Developing a Meta Model and Coupling of Freight Demand Models for the CEP-Sector

Institute for Transport Studies
Department of Civil Engineering, Geo and Environmental Sciences
Karlsruhe Institute of Technology

Master's Thesis

of

Simon Haug

Matriculation Number 2427358

Reviewer:
Prof. Dr.-Ing. Peter Vortisch

Second Reviewer:
Prof. Dr. Ralf Reussner

Advisors:
M.Sc. Jelle Kübler
Dr.-Ing. Erik Burger

Date of Submission: 15.05.2024
Karlsruhe

Declaration

I declare that I have developed and written the enclosed thesis without the assistance of others, and have not used sources or means without declaration in the text. External assistance includes paid services of third parties, e.g. Internet agencies, which check the written version for spelling and / or punctuation, grammar, formulation, logic, plausibility, sentence structure, sense or style. This work has not yet been submitted to another examination institution neither in the same nor in a similar way and has not yet been published. This thesis was carried out in accordance with the Rules for Safeguarding Good Scientific Practice at Karlsruhe Institute of Technology (KIT). I hereby grant the Institute for Transport Studies the right to use this thesis. This includes in particular the further use of the acquired knowledge and results in other works or publications by employees of the Institute for Transport Studies. I also grant the reference library of the Institute the right to use this thesis.

Karlsruhe, 15.05.2024

Simon Haug

Karlsruher Institut für Technologie

Fakultät für Bauingenieur-, Geo- und Umweltwissenschaften

Institut für Verkehrswesen
Leiter: Prof. Dr.-Ing. Peter Vortisch

Kaiserstraße 12
Geb. 10.30, 3. OG
76131 Karlsruhe
www.ifv.kit.edu

Aufgabenstellung für die Masterarbeit

von

Simon Haug

Matrikelnummer **2427358**

mit dem Titel:

Entwicklung eines Meta-Modells zur Kopplung von Güterverkehrsmodellen für den KEP-Sektor

Englischer Titel: Developing a Meta Model and Coupling of Freight Demand Models for the CEP-Sector

Problemstellung

In den letzten Jahren ist die Nachfrage im Onlinehandel deutlich gestiegen und zur Erfüllung dieser Nachfrage auch der Lieferverkehr. Dieser Trend wurde durch die COVID-19 Pandemie zusätzlich verstärkt. Bei der Paketlieferung auf der letzten Meile werden heutzutage größtenteils Kleintransporter mit einem zulässigen Gesamtgewicht bis 3,5t eingesetzt. Diese konventionellen Lieferfahrzeuge haben einen hohen Emissionsausstoß und verursachen vor allem in dicht bebauten Gebieten Platzprobleme.
Daher sind sowohl Verkehrsplaner als auch KEP-Dienstleister daran interessiert neue Lieferkonzepte einzuführen. Unter anderem wurden Ansätze wie Paketschließfächer, Lastenräder, Micro-Hubs und Cargo-Trams getestet und teilweise auch schon etabliert.
Um die Auswirkungen solcher neuen Maßnahmen auf die Verkehrsnachfrage und den Emissionsausstoß zu prognostizieren, werden Güterverkehrsmodelle in der Verkehrsplanung verwendet. Verkehrsplaner und politische Entscheider können damit verschiedene Maßnahmen vor deren tatsächlichen Implementierung bewerten und vergleichen.
Um eine detaillierte Analyse der Güter- und Verkehrsnachfrage, sowie der einzelnen Akteure und deren Beziehungen zu ermöglichen, sind agentenbasierte Modelle im Fokus aktueller Forschung. Es wurden bereits diverse Modelle entwickelt, welche den Güterverkehr und den Privatverkehr integriert betrachten, um die Interaktionen beider Bereiche abzubilden. Dabei hat jedes dieser Modelle spezifische Anwendungsfälle und verschiedene Stärken und Schwächen. Bedingt durch die hohe Komplexität der Güterverkehrsdomäne wurde noch kein vollumfassendes, agentenbasiertes Modell entwickelt.

Aufgabenstellung

In dieser Arbeit soll ein Ansatz untersucht werden, der es erlaubt, die Vielfalt an Funktionen mehrerer agentenbasierter Güterverkehrsmodelle zu kombinieren und integriert nutzbar zu machen. Dazu sollen Modelle zur Abbildung des Güterverkehrs (mit Fokus auf den KEP-Sektor) analysiert und verglichen werden, um gemeinsame Konzepte und Strukturen zu extrahieren. Basierend darauf sollen Modelltransformationen definiert werden, mit welchen eine beispielhafte Kopplung zweier solcher Modelle umgesetzt werden soll. Für diese Arbeit sollen die Güterverkehrsmodelle logiTopp und MATSim-Freight genauer untersucht werden.

Karlsruher Institut für Technologie

Als Grundlage der Arbeit sollen zunächst verschiedene agentenbasierte Güterverkehrsmodelle recherchiert werden. Hierbei sollen die umgesetzten Konzepte und die Anwendungsgebiete der Modelle herausgearbeitet werden. Weiter sollen die relevanten Grundlagen aus dem Bereich der Metamodellierung und Modelltransformation zusammengefasst werden. Unter anderem sollen dabei auch Metriken und Kriterien recherchiert werden, die zur Bewertung von Metamodellen und Modelltransformationen angesetzt werden können.

Basierend darauf sollen (mindestens) die beiden Güterverkehrsmodelle logiTopp und MATSim-Freight detailliert analysiert und verglichen werden. Gemeinsame Konzepte und Strukturen sollen in Form eines allgemeinen Metamodells extrahiert werden. Auch die Unterschiede der Modelle sollen herausgearbeitet und diskutiert werden. Neben der Definition des Metamodells sind die Modelltransformationen zwischen Metamodell und den untersuchten Modellen zu definieren. Für die gefundenen Modell-Unterschiede sollen mögliche Lösungen für eine konsistente Modellierung im Metamodell erarbeitet und Limitierungen aufgezeigt werden.
Mit den entwickelten Modelltransformationen soll anschließend eine beispielhafte Kopplung der beiden Modelle logiTopp und MATSim-Freight umgesetzt werden. Welche Anwendungsfälle sich für eine Kopplung eignen, soll in dieser Arbeit erörtert werden. Hierbei sollen auch verkehrliche Kenngrößen evaluiert werden.

Abschließend soll das entwickelte Metamodell, die Modelltransformationen und die Kopplung evaluiert werden. Dazu sollen die recherchierten Bewertungskriterien aus der Literatur herangezogen werden unter Beachtung ihrer spezifischen Anwendbarkeit. Weiter können eigene Metriken zur Evaluation ergänzt werden. Zur Evaluation der Kopplung soll diese außerdem auf ein vom IfV bereitgestelltes Modell der Region Karlsruhe angewendet werden. Zur Bewertung der Anwendung sollen verkehrliche Kenngrößen auf Plausibilität geprüft werden.

Die Arbeit ist gebunden (DIN A4) in zweifacher Ausführung einzureichen und nach Möglichkeit ist die Formatvorlage des Instituts für Verkehrswesen zu verwenden.

Betreuer: Prof. Dr.-Ing. Peter Vortisch
Prof. Dr. Ralf Reussner

Betreuende wissenschaftliche Mitarbeiter: M.Sc. Jelle Kübler
M.Sc. Lukas Barthelmes
Dr.-Ing. Erik Burger
M.Sc. Jan Ellmers

Karlsruhe, den 30.10.2023

Peter Vortisch

Prof. Dr.-Ing. Peter Vortisch

ausgegeben: 15.11.2023
abzugeben: 15.05.2023

Abstract

CEP service providers and transport planners are interested in optimizing parcel transport processes and developing new concepts for this purpose. Freight transport models play an important role here, as they allow evaluation and comparison of a variety of new approaches at an early stage and at low costs. A large number of freight transport models have already been developed for this purpose, each of which is tailored to specific scenarios and research questions. However, these models generally use different data models and input data, which requires a major manual effort to simulate scenarios in different models. Automated coupling of freight transportation models has so far only been developed for a few individual pairs of models.

This work presents an approach based on model-driven engineering for the automatic coupling of multiple freight transport models with the aim of combining their variety of functions and making them usable in an integrated manner. For this purpose, a common metamodel for the domain was developed, which serves as a central exchange point for the transformation between the models. This metamodel describes the domain's central and common concepts, encapsulates their variability, and defines a common view. As part of the evaluation of the approach, an example model from the Rastatt area in Germany was coupled between the logiTopp and MATSim-Freight freight transport models. It could be shown that the elementary properties of the source model were correctly transferred to the coupling's target model.

Zusammenfassung

KEP-Dienstleister und Verkehrsplaner sind daran interessiert, die Prozesse im Transport von Paketen zu optimieren und dafür neue Konzepte zu entwickeln. Dabei spielen Güterverkehrsmodelle eine entscheidende Rolle, da sie es ermöglichen, eine Vielzahl von neuen Ansätzen frühzeitig und kostengünstig zu bewerten und zu vergleichen. Dazu wurden bereits eine Vielzahl von Güterverkehrsmodellen entwickelt, die jeweils auf spezifische Anwendungsfälle und Fragestellungen zugeschnitten sind. Jedoch verwenden diese Modelle in der Regel unterschiedliche Datenmodelle und Eingabedaten, was einen hohen manuellen Aufwand erfordert, um Szenarien in verschiedenen Modellen zu simulieren. Eine automatisierte Kopplung von Güterverkehrsmodellen wurde bisher nur für vereinzelte Paare von Modellen entwickelt.

Diese Arbeit präsentiert einen Model-Driven Engineering basierten Ansatz zur automatischen Kopplung beliebig vieler Güterverkehrsmodelle mit dem Ziel, die Vielfalt an Funktionen zu kombinieren und integriert nutzbar zu machen. Dazu wurde ein allgemeines Metamodell für die Domäne entwickelt, das als zentraler Austauschpunkt für die Transformation zwischen den Modellen dient. Dieses Metamodell beschreibt zentrale und allgemeingültige Konzepte der Domäne, kapselt deren Variabilität und definiert eine allgemeine Sichtweise. Im Rahmen der Evaluation des Ansatzes wurde exemplarisch eine Kopplung eines Beispielmodells aus dem Raum Rastatt, Deutschland, zwischen den Güterverkehrsmodellen logiTopp und MATSim-Freight durchgeführt. Zudem konnte gezeigt werden, dass elementare Eigenschaften des Quellmodells korrekt in das Zielmodell der Kopplung übertragen wurden.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

The transportation of parcels has seen a significant increase in recent years. In Germany alone, the number of parcels handled by courier, express, and parcel (CEP) service providers reached 4.51 billion in 2021, marking a substantial increase of around 2 billion parcels compared to 2012 [12]. This surge in parcel volumes presents major challenges for CEP service providers (CEPSPs) and transport planners, particularly in densely populated areas where last-mile delivery by small vans contributes to traffic congestion, emissions, and space constraints. Addressing these challenges requires further optimization of CEP processes and the development of innovative parcel delivery solutions, such as parcel lockers, drones, cargo bikes, and integrating public transportation into transport chains.

Freight transport models are valuable tools for developing and evaluating these solutions. They enable the simulation of various approaches before implementation, allowing us to analyze their impact on factors like efficiency, costs, and emissions. While various freight transport models have been developed, current research primarily focuses on agent-based approaches. In these models, each relevant entity in the domain, such as delivery vehicles or dispatchers, is represented as an autonomous agent that makes individual decisions and interacts with infrastructure and other agents.

In the meantime, various freight transport models have been developed. Each freight transport model focuses on different aspects of the domain and is tailored to specific research questions. Thus, each model has its strengths and weaknesses. No fully comprehensive model has been developed due to the domain's complexity and the multitude of research questions to be addressed.

Therefore, leveraging the capabilities of different freight transport models is essential for gaining a comprehensive understanding of new approaches in the freight transport domain. However, this process is labor-intensive, as researchers must model scenarios separately in multiple freight transport models and, often manually or with specialized tools, transfer interim results between models. This requires significant effort and poses challenges to data integrity and result correctness.

To streamline this process, there is a need for tools that facilitate coupling or automatic data exchange between various agent-based freight transport models at different stages of the modeling process. It would also benefit this process to develop a common view of this domain shared by as many freight transport models as possible. This common view also can serve as the foundation for the coupling and data exchange.

1.2 Goal of the Thesis

This thesis aims to develop, prototypically implement, and evaluate a concept that facilitates the usage of multiple freight transport models in an integrated workflow. And thus utilizing the diverse features and strengths of various freight transport models.

This thesis pursues an approach based on the idea of a common metamodel for the freight transport domain. This metamodel serves as a unified understanding of the domain and acts as a central point for coupling and data exchange among different models. Moreover, metamodeling allows the application of tools and concepts from the field of model-driven engineering. This common metamodel represents a further key outcome of this thesis.

To achieve these objectives, an analysis of existing freight transport models will be conducted to identify commonalities and similarities in their concepts and structures. Subsequently, based on this analysis, the common metamodel and a concept for integrating diverse freight transport models will be developed. This framework will be applied to the logiTopp [31, 49] and MATSim-Freight [53, 54, 61, 62] models through a prototypical implementation. This implementation will serve as the foundation for evaluating the efficacy of the developed concept.

1.3 Structure of the Thesis

Chapter 2 introduces relevant foundations for this work. We present the idea of modeling, metamodeling, and concepts of model-driven engineering. After that, we briefly introduce software product line engineering and introduce freight transport models by presenting the general structure and existing models. Further, we present related work in the field of coupling transport models and models in general.

In Chapter 3, we analyze and compare the metamodels, i.e., the used domain models, of several freight transport models, mainly of logiTopp and MATSim-Freight. The developed concept for coupling multiple freight transport models is presented and discussed in Chapter 4. In Chapter 5, we then present and discuss the developed

common metamodel and the required transformations.

A prototypical implementation of the concept is briefly introduced in Chapter 6. In Chapter 7, we evaluate our work with a brief case study and testing of preservation of elementary properties between coupled models.

Finally, in Chapter 8, we close the thesis with a conclusion, which summarizes the results of our work and gives an overview of possible future work.

2 Foundations

This chapter introduces the foundations on which this work is based. We begin by exploring the fundamental principles of model-driven engineering, including metamodelling and model transformations, in Section 2.1. Next, we provide a brief overview of some terms and concepts of software product line engineering in Section 2.2. After that, we delve into the foundational aspects of freight transport models and examine existing models, such as logiTopp and MATSim-Freight, in Section 2.3. Finally, we provide a brief overview of related work in the field of coupling of freight transport models in Section 2.4.

2.1 Model-Driven Engineering

Model-driven engineering (MDE) is a paradigm that considers models as first-class entities throughout the entire development and operation of (software) systems [15]. The central idea of this paradigm is to raise the level of abstraction through the use of models and then apply generative and transformational techniques to reduce complexity and increase the level of automation. MDE uses a set of well-defined practices such as metamodeling and model transformation [25].

Model-Driven Development (MDD) and Model-Driven Software Development (MDSD) are often used interchangeably concerning MDE. Both refer to applying MDE methods in software development. While encompassing these practices, MDE incorporates a broader range of concepts, tools, and approaches associated with using models in engineering practices beyond software development.

This section introduces key concepts of MDE used in this work. Section 2.1.1 describes the common understanding of the term model in the context of MDE. Next, Section 2.1.2 presents the concept of metamodelling, followed by an introduction to the used framework for it (see Section 2.1.3). The concept of model transformations is introduced in Section 2.1.4.

2.1.1 Model Term

The term model has various meanings in different scientific contexts. In the context of MDE, a model is an abstract representation of a (software) system. Which describes only certain aspects of a system. The high degree of abstraction and the focus on a reduced set of aspects reduce the complexity of understanding, handling, and maintaining the model.

A common and in the context of MDE suitable characterization of the term model is provided by Stachowiak [55], who describes three properties every model fulfills:

- Representation property: Models always represent some originals. These originals can be natural or artificial nature and even be models themselves.
- Reduction property: Models do not include every property of the represented original. Instead, they consist only of a reduced subset of properties relevant to the use case.
- Pragmatic property: Models are designed for a specific context. It acts as a substitution function tailored for a specific user, during a specific period, and for a certain set of operations.

Section 2.1.3 gives a brief example of a model that falls into this definition.

2.1.2 Metamodelling

A metamodel describes the allowed structure of models, encompassing their available elements, attributes, relationships, constraints, and other modeling rules. Models that adhere to this structure are recognized as instances of the metamodel, establishing a direct relationship. A metamodel is a model itself. Metamodels commonly undergo formal specification. Stahl et al. [56] propose that a comprehensive metamodel specification should encompass an abstract syntax, one or more concrete syntaxes, static semantics, and dynamic semantics. These individual components are elaborated

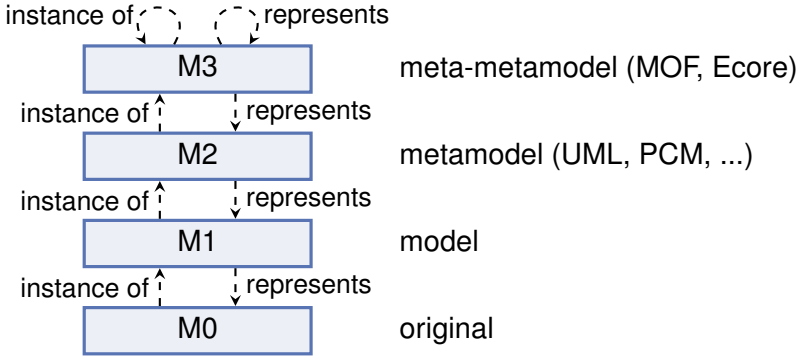


Figure 2.1: Layered metamodel architecture of MOF with four layers.

upon as follows:

- Abstract Syntax: The abstract syntax of a metamodel defines the elements comprising a model, along with their properties and relationships. This description remains independent of any specific concrete representation.
- Concrete Syntax: A metamodel encompasses one or more concrete syntaxes that detail how model elements and their relations are represented. These syntaxes can take various forms, such as graphical or textual representations, or may specify a particular file format.
- Static Semantics: The static semantics of a metamodel encapsulate all constraints and modeling rules that cannot be articulated within the abstract syntax.
- Dynamic Semantics: The dynamic semantics express the meaning behind the model elements and their relationships. Often expressed in natural language, dynamic semantics can also manifest through transforming the model into another formal language with predefined semantics, such as code, an automaton, or petri-networks.

There are two elementary relations a model has to other models. Every model *represents* some kind of original, which can be a real object or a model itself. The *instance of* relation establishes a crucial link between a model and its metamodel, stating that the model conforms to the structure and constraints defined within its associated metamodel. The metamodel of a metamodel is called a *meta-metamodel*. Theoretically, any number of modeling levels can be created following this pattern. However, usually, there is, at some point, a self-describing modeling level, which ends this cascade. In most cases, the number of modeling levels is limited to four.

An example of a self-describing meta-metamodel is the Meta-Object Facility (MOF) [43]. Which is used by the UML standard [45] in a four-layer hierarchy. Figure 2.1 shows the layered metamodel architecture of MOF with four layers. MOF uses the Object Constraint Language (OCL) [42] to express the static semantics of its metamodels. OCL allows declarative specifications constraints and invariants on classes, which can then be evaluated in their instances. It can also be used as a query language, which is used, for example, in the context of model transformations.

2.1.3 Eclipse Modeling Framework

The Eclipse Modeling Framework (EMF) [57] is an Eclipse-based modeling framework and code generation facility for building tools and other applications based on metamodels. Its core component is the meta-metamodel Ecore, which is compatible with the Essential MOF (EMOF) subset of MOF [43], which is tailored to representing object-oriented constructs. Figure 2.2a shows a simplified subset of the Ecore

meta-metamodel.

EMF provides various tools and extensions for the handling of Ecore-based metamodels. These metamodels and their instances can be created and modified through UI editors or a reflective Java API. EMF also provides tools for generating Java code and persisting models.

Figure 2.2 presents a simple illustration of an EMF-based metamodel and model, demonstrating the concepts discussed earlier. We have refrained from showing the original family (level M0).

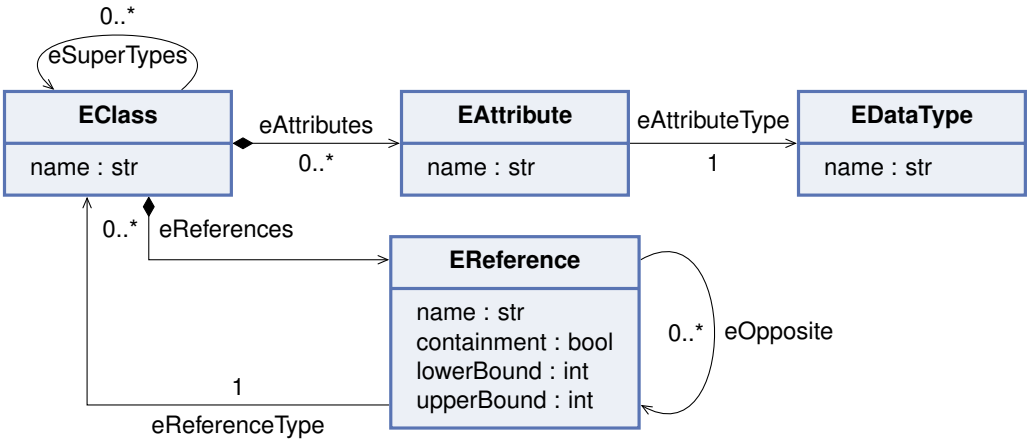
The diagram depicts in Level M3 a simplified subset of the Ecore meta-metamodel, showcasing how classes with attributes of specific data types and references to other elements are described. The meta-metamodel's self-describing property is evident, as all elements used in the meta-metamodel are described by the meta-metamodel itself. For instance, the meta-metamodel describes classes (*EClass*) with attributes (*EAttribute*) of a type (*EDataType*) and references (*EReference*) to other classes, showcasing its ability to represent its own structure and constraints.

Moving down to Level M2, we encounter the metamodel for describing families, which is an instance of the Ecore meta-metamodel. For example, the family metamodel includes an *EClass* named *Family* with an *EAttribute lastName* of type String (*str*) and multiple containment *EReferences* to the *EClass Member*, outlining the structure of family models.

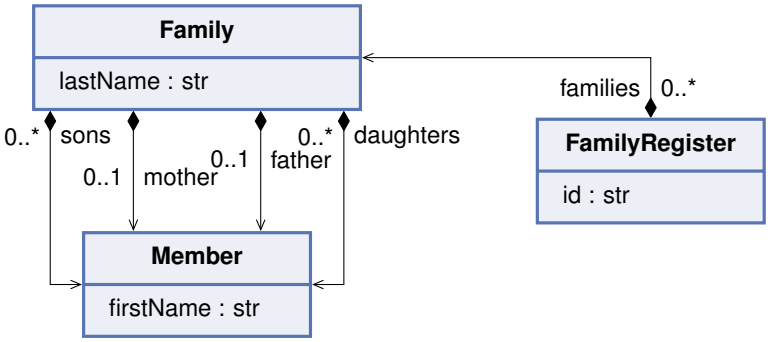
At Level M1, we find the actual model describing the Smith family. While not explicitly shown, it is clear that the model represents a real family (representation property). Note that this model does not describe every property of the family and its members (reduction property). For instance, there is no description of the hair color of family members. This omission may occur because the model is tailored for a specific context (pragmatic property). Thus, the missing property is irrelevant in this context. It potentially serves as part of a larger metamodel describing insurance contracts for families. In this context, details like hair color may be deemed irrelevant.

2.1.4 Model Transformations

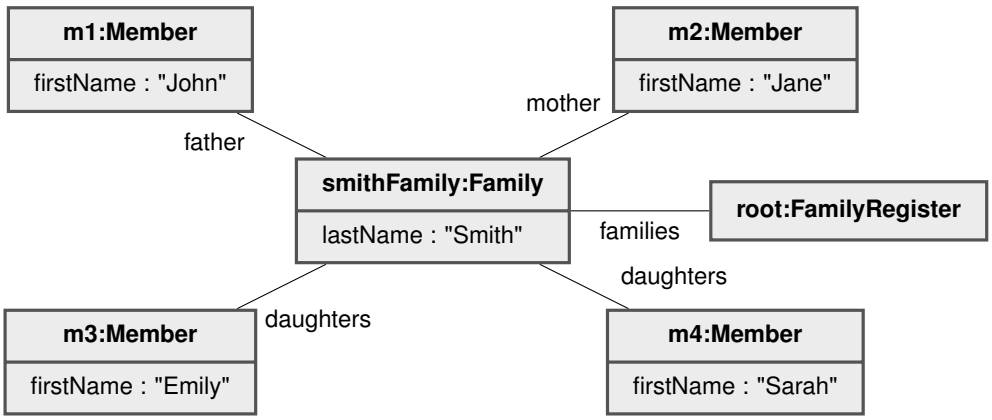
Kleppe et al. [28] define model transformations as "the automatic generation of a target model from a source model, according to a transformation definition." A transformation definition is a set of rules describing how source model elements are transformed into a target model. Model transformations are a key technique of MDE. They can be employed for various tasks such as generating lower-level models, creating query-based views, performing model evolution tasks like refactorings, and mapping and



(a) M3: A reduced and simplified subset of the Ecore meta-metamodel (adapted from [57]).



(b) M2: A Ecore based metamodel for the description of families (no eOpposite references are shown).



(c) M1: Model of a family that is a instance of the family metamodel (Figure 2.2b).

Figure 2.2: Example of an EMF-based model hierarchy describing families.

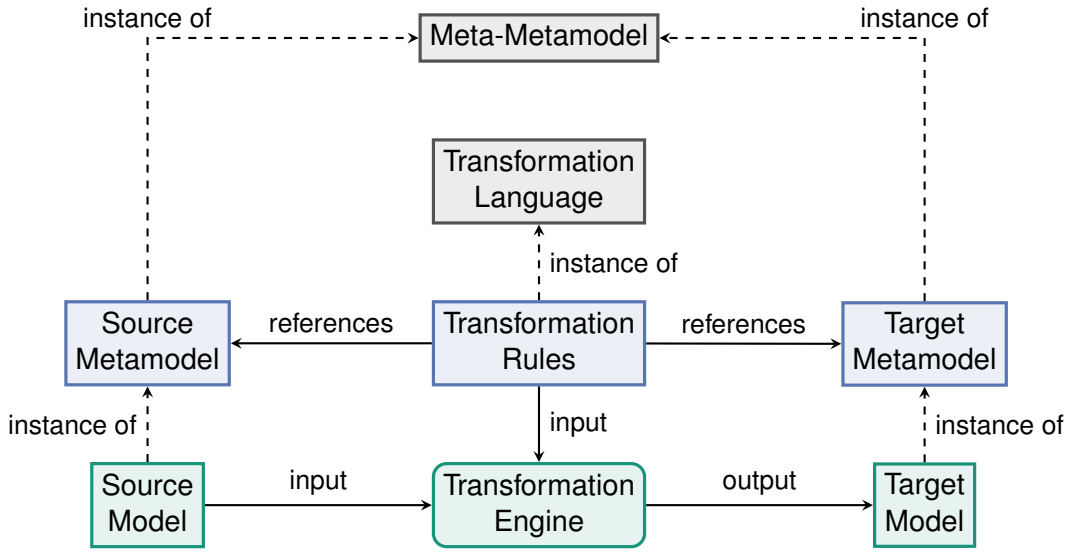


Figure 2.3: Basic schema for model transformations.

synchronizing among models [17].

Figure 2.3 provides an overview of the core principle of model transformations, illustrating the straightforward scenario of transferring a source model to a corresponding target model within the same technical space. Both the source and target models adhere to their respective metamodels, and the transformation is defined by a set of rules specific to these metamodels. The transformation rules are applied by a transformation engine on the input model to execute a transformation.

Mens and Van Grop [38] present a taxonomy that categorizes model transformations along five dimensions:

- Number of source and target models: Transformations may involve multiple source and target models.
- Technical space: This refers to the model management framework used, determined by the meta-metamodel of the source and target models. Models can reside in the same or different technical spaces.
- Endogenous vs. exogenous transformations: Endogenous transformations operate on instances of the same metamodel, while exogenous transformations map between different metamodels.
- Horizontal vs. vertical transformations: Horizontal transformations occur within the same level of abstraction, while vertical transformations map between different levels of abstraction.
- Syntactical vs. semantical transformations: Syntactical transformations consider only the syntax of models, while semantical transformations also incorporate their semantics. Code generation exemplifies syntactical transformations, while

refactorings and optimizations are semantical transformations.

Mens and Van Gorp [38] also identify several key characteristics of model transformations, such as their complexity, level of automation (whether manual input is required for execution), and which aspects of the source model are preserved in the target model.

Another categorization distinguishes between Model-to-Model (M2M), Model-to-Text (M2T), and Text-to-Model (T2M) transformations. M2M transformations convert one model to another, conforming to explicitly defined metamodels. M2T transformations produce textual artifacts, such as code or documentation, from models, whereas T2M transformations, or reverse engineering, generate models from textual artifacts. While textual artifacts are often considered separate from models, they mostly conform to the model term and an, often implicitly given, metamodel in this context.

Many different languages exist to specify model transformations and express transformation rules. They can be divided into declarative, imperative, and hybrid languages. Where hybrid languages combine declarative and imperative language constructs.

Declarative model transformation languages specify the desired output rather than procedural steps. They allow to express relations between the source and target models, enabling developers to concentrate on the "what" rather than the "how" of the transformation process. Thereby, details such as rule execution order, target model creation, and source model navigation are hidden. One benefit of declarative languages is their ease in creating bidirectional transformations. Because relations inherently express both forward and backward transformation directions. A commonly used language family for model transformations, compatible with MOF, is QVT (Query/View/Transformation) [44] that provides both a declarative (QVT-R) and imperative (QVT-O) language.

For more information about model transformations, especially regarding their specification and implementation, we refer the reader to the works of Czarnecki and Helsen [17] and Di Ruscio et al. [20].

2.2 Software Product Line Engineering

This section gives a brief overview of some terms and concepts of the field of Software Product Line Engineering (SPLE) used in this work. Pohl et al. [47] define SPLE as a "paradigm to develop software applications [...] using platforms and mass customization". It involves concepts for managing the commonalities and variabilities across the products of a software product line.

The terms *variability subject* and *variability object* are introduced to describe variability. Variability subjects are variable items or properties in the analyzed system, while

variability objects are the particular instances of the variability subject [47]. In the context of software product lines, a variability subject is called a *variation point* that has multiple *variants*. In contrast to variability subjects and objects, variation points and their variants are associated with contextual information and domain artifacts [47].

Modeling variability is an essential technique for documenting and managing the variability of software product lines. A common approach is using feature models, which are graphically represented as feature diagrams [6]. These models describe relationships between features and define valid combinations of features. A feature model is essentially a hierarchical tree structure where each element has specific semantics, as described by Apel et al. [6]:

- Child features: These can only be selected if their parent is selected.
- Optional and mandatory features: Mandatory features must be selected if their parent is selected, while optional features may or may not be selected.
- Concrete and abstract features: Concrete features are tied to implementation artifacts, whereas abstract features serve only for structural organization.
- Or group: At least one child feature within this group must be selected if the feature is selected.
- Alternative group: Exactly one child feature within this group must be selected if the feature is selected.
- Cross-tree constraints: These allow expressing relations between features that cannot be captured within the hierarchical structure of the tree. Propositional formulas over the features are used to express these relations.

Figure 2.4 depicts the graphical notation of feature diagrams. A set of selected features from a feature model is called a *feature configuration*.

Software product lines have their own engineering processes. One such process, described by Apel et al. [6] and illustrated in Figure 2.5, is feature-oriented software product line engineering. This process is structured along two dimensions. Firstly, it distinguishes between *domain engineering* and *application engineering*. Domain engineering is development for reuse, which comprises analyzing the product line's domain and developing reusable artifacts. Application engineering then deals with developing an application adapted to a customer's needs by analyzing those needs and creating the application from the reusable artifacts. Secondly, it separates the *problem space* from the *solution space*. The problem space encompasses the stakeholder's viewpoints, emphasizing domain problems, requirements, and abstractions represented by features. In contrast, the solution space caters to developer needs, focusing on design, implementation, and validation using technical terminology, aiming to enable systematic reuse. This distinction results in four task clusters:

- Domain Analysis: This is a form of requirements engineering and involves defining

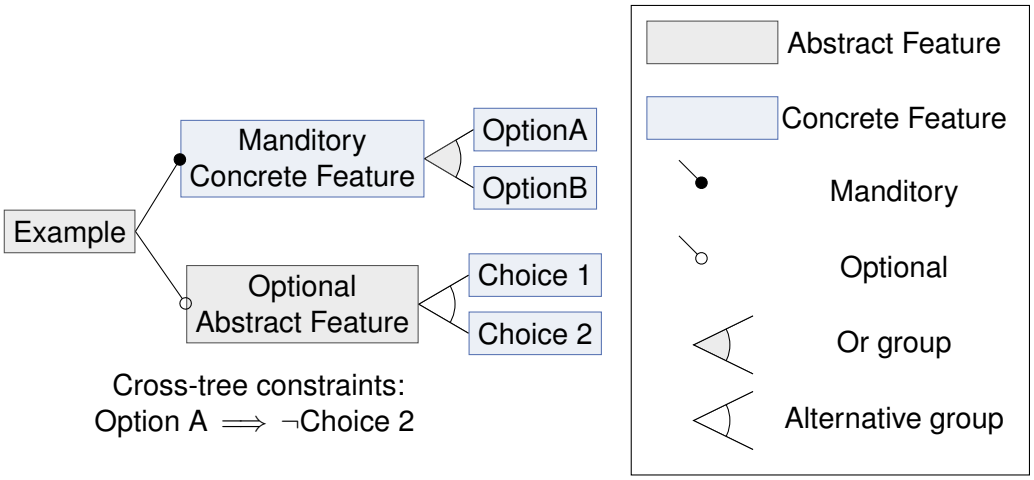


Figure 2.4: Legend and an example of a feature diagram.

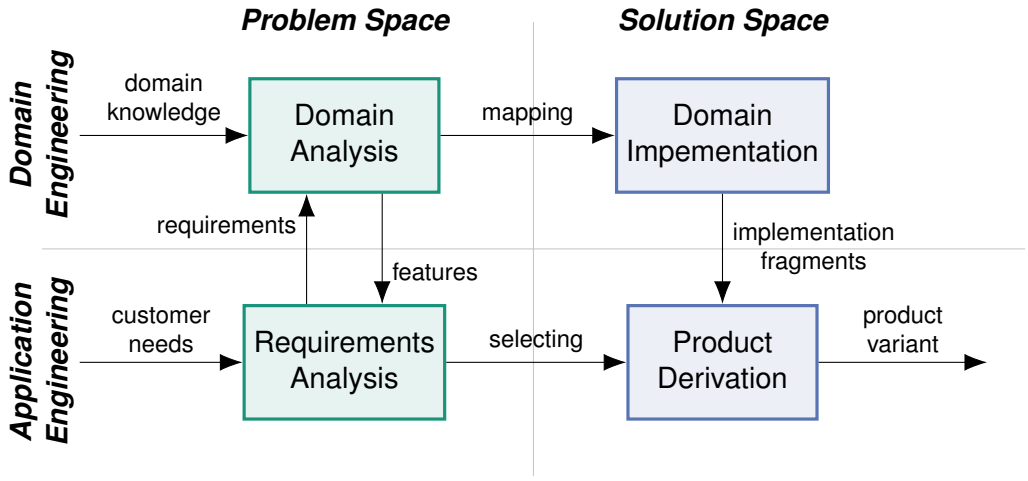


Figure 2.5: Overview of an engineering process for software product lines (adapted from [6]).

the scope of the product line by analyzing domain problems and requirements. It results in a feature model representing the variability of the software product line.
- Requirements Analysis: Here, the needs of a specific customer are analyzed and mapped to a feature configuration of the feature model. Any missing features are fed back into the domain analysis.
- Domain implementation: This encompasses the development of reusable artifacts according to the feature model. It also includes deciding which concepts are applied to implement the variability.
- Product derivation: In this step, the reusable artifacts are assembled according to the selected feature configuration to derive a product variant.

2.3 Freight Transport Modelling

Freight transport is a complex domain with many stakeholders and complex interactions among them. These interactions include the exchange of goods between producers and consumers within the framework of supply and demand mechanisms intricately entwined with the realization of supply chains. Additionally, there is the market for providing logistic services and capacities and the process of orchestrating logistic solutions, such as mode choice and trip generation. Another crucial interaction involves the utilization of existing traffic infrastructure. This is usually a shared resource used by both members of the transport market and entirely different entities, such as private persons.

Freight transport models describe these processes and interactions. According to Tavasszy and De Bok [58], their main function is "to provide comprehensive information to all stakeholders about the current and expected performance of the system under different future social, economic and technological scenarios: the models are used to assess the effects of these scenarios." To handle this complexity, freight transport models usually focus on a subset of the interactions and processes of the freight transport domain (reduction property). They are designed to assess specific effects of the evaluated scenarios (pragmatic property) [58]. This, together with advances in computer science and modeling, is the reason why a large number of freight transport models have been developed in the past. To give a brief overview of possible approaches, the topology for freight transport models proposed by Thaller et al. [60] is presented in the following. They characterize freight transport models by the following characteristics:

- **Level of Aggregation**: *Aggregated* models segment the study area into traffic analysis zones. The demand generation and subsequent steps are then calculated per traffic analysis zone. In *disaggregated* models the behavior of individuals or behavior-homogenous groups considered for the demand calculation.

An increasingly popular variant of disaggregated models is agent-based modeling, in which every participant in the freight market is considered an individual agent. The decisions, interactions, and actions of each agent are simulated separately.

- **Scale of Analysis:** *Macroscopic* models make use of aggregated data and behaviour-homogenous groups. While *microscopic* models calculate the demand of individual actors. *Mesoscopic* models combine both approaches.
- **Reference Values**: The Reference value is the first occurring value of a model. This is an indicator of the internal structure and implies which decisions are modeled implicitly or explicitly. Reference values can either be *freight or goods flow based*, *vehicle based*, or of a *hybrid* form.
- **Study Area**: Categories for the size of the study area are *national*, *regional*, *urban*, or *local*. The spatial resolution of the model typically depends on the size of the study area (which is the original category proposed by Thaller et al. [60]).

Further characteristics are the internal architecture and structure of the models (see Section 2.3.1) as well as the used formal modeling methods. Typical modeling methods include statistical analysis, statistical simulations, distribution methods, optimization, equilibria, and temporal simulations. Based on the assignment of this work, we mainly focus on disaggregated, agent-based, macroscopic models with a focus on regional study areas.

Section 2.3.1 continues with an introduction to the architecture and structure of freight transport models. After that, existing freight transport models are presented in Section 2.3.2, starting with the presentation of the models which are examined in this work in more detail, followed by a short literature review on other existing freight transport models. For a broader overview of the field of freight transport and city logistics and their modeling and simulation, we refer the reader to the work of Marcucci et al. [37], Tavasszy and Jong [59], and Thaller et al. [60].

2.3.1 Model Structure

Due to the wide array of freight transport models, there is no universal model structure that encapsulates all of them. Thaller et al. [60] attempted to outline and generalize freight transport modeling steps and structure, which is depicted in Figure 2.6 and elaborated further. This attempt is noteworthy for its simplicity and high-level perspective, making it comprehensible. The 9-step modeling adapt the well-known 4-step approach used for conventional traffic models. Based on this model, the key modeling steps and their relation to logistic decisions can be identified. It's important to recognize that not every freight transport model integrates all of these modeling steps and logistic considerations.

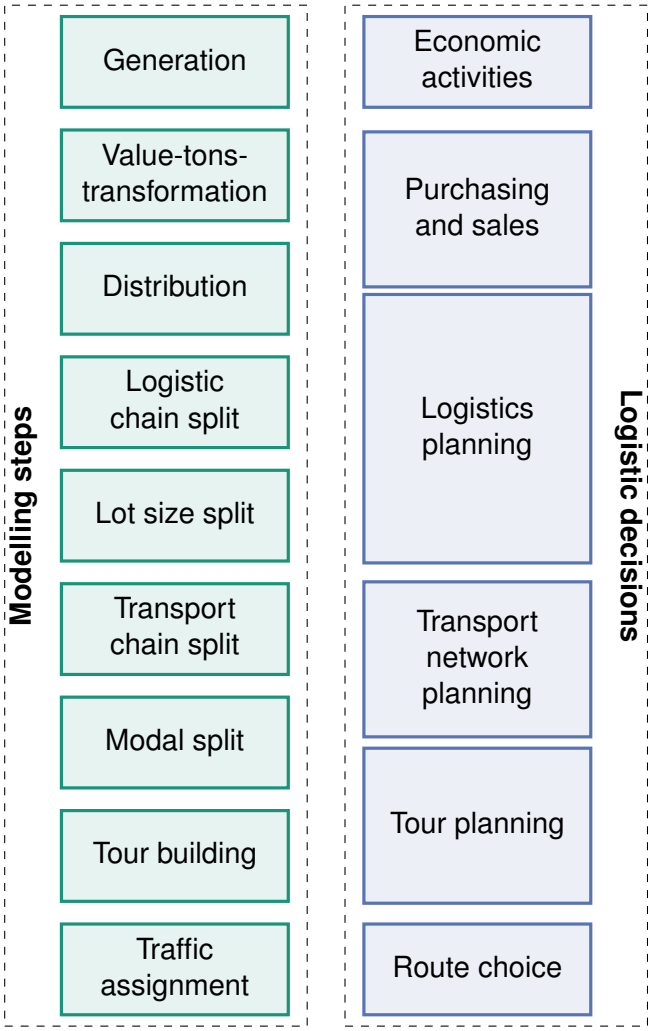


Figure 2.6: Freight transport modelling steps and logistic decisions (adapted from [60]).

The model starts with the generation of the freight demand (*Generation*), which can be seen as a model of the commodity market. In this step, the demand is represented in a monetary or other abstract representation, which is transformed into physical units such as volume and weight in the *value-tons-transformation*. Now, the freight transport volume is distributed spatially in the study area (*Distribution*), resulting in the amount of freight that is transported between a single origin and destination or origin and destination areas. In the *logistics chain split*, the freight flows are assigned to several logistic chains and split into single deliveries (*Lot size split*). After that, the freight flows are assigned to the available transport chains (*Transport chain split)*. For every leg of a transport chain, the used transport mode is chosen (*Modal split*). In the last two steps, the single delivery trips are merged into complex tours (*tour building*), and the resulting traffic is allocating capacities of the infrastructure network (*traffic assignment*). All these presented steps model logistic decisions, which are shown to the right of the corresponding modeling step in Figure 2.6.

However, this 9-step model does not take into account some common aspects of freight transport models. Due to the step-wise approach, only information from previous steps is taken into account in a modeling step, never information from subsequent decisions. For example, the tour building step does not consider the actual traffic volumes of the transport network, which are determined in the traffic assignment step. Thus, the model lacks representation of iterative processes. In addition, some agent-based approaches are difficult to describe using this approach, as the agents often live in the simulation environment and make decisions based on interactions with other agents in a more or less arbitrary order.

2.3.2 Existing Freight Transport Models

LogiTopp

LogiTopp [31, 49] is a freight transport model focusing on modeling parcel orders and simulation of last-mile deliveries. It is an extension of the activity and agent-based travel demand simulation model mobiTopp [35, 36]. Both mobiTopp and logiTopp are available as open-source projects on GitHub [30, 33]. We start with a short introduction to the concepts and structure of mobiTopp and then present its extension, logiTopp.

In mobiTopp, every person in the generated population is an agent with specific attributes, such as age, associated household, car ownership, or tickets for public transport services. For every agent, an activity schedule is generated based on its attributes and then executed dynamically. The execution of the activity schedule includes the mode and destination choice for each trip. The simulation does not include a traffic assignment, so there is no notion of interaction between agents through the usage of the transport infrastructure, e.g., by traffic jams. The used travel times are based on externally precalculated origin-destination matrices. However, agents can interact indirectly through the availability or non-availability of cars in the household context, e.g., if another household member uses the car while the agent makes its next mode choice. There also exist several extensions, such as the simulation of carsharing and ridesharing. MobiTopp supports simulation periods of up to one week.

Figure 2.7 shows the structure of mobiTopp. The long-term module covers long-term aspects of the system, such as population synthesis, assignment of fixed destinations (e.g., workplaces), generation of activity schedules, and determination of available mobility tools (e.g., car and bike ownership, commuter tickets, or car sharing membership). These properties are fixed and are the foundation for the simulation of the travel behavior in the short-term module. The short-therm module mainly consists of a destination and a mode choice model, which is supported by the mode availability model that keeps track of available transport modes for a given agent at a given time. The activities are

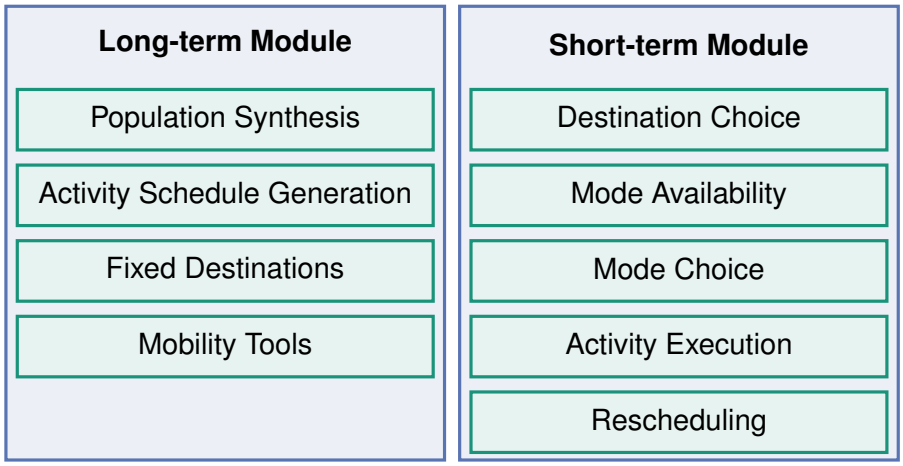


Figure 2.7: Structure of mobiTopp (adapted from [35]).

then executed, and the rescheduling model adapts the activity schedule to take the experienced travel times into account and adapt the schedule. Due to this modular structure and the use of object-oriented programming, all these models can easily be exchanged or adapted. Making mobiTopp a very flexible framework.

LogiTopp extends mobiTopp by the simulation of last-mile parcel deliveries. One of the distinctive features of logiTopp is the detailed simulation of the delivery of parcels. Parcels can be delivered to an agent's workplace, home, or a selected packstation. The model takes into account whether the receiving agent, another household member, or even neighbors are at home during the delivery attempt. If that is not the case, the parcel will be taken back to the depot and delivered the next day or rerouted to a packstation. The model has been extended to also simulate packages sent or received by businesses [9] and takes contractual relations between businesses and CEPSPs [32] into account. An overview of the supported producer-consumer relations is shown in Figure 2.8. CEPSPs function as entry and exit points for parcels entering or leaving the study area. They serve as either the starting or ending points of a parcel's journey within the study area. This means that when a parcel is sent within the study area and handled by a CEPSP, it entails two trips: one from the origin to the CEPSP and another from the CEPSP to the destination. At the current stage of development, private persons cannot send parcels. The trips of a CEPSP to pick up parcels directly or from a post office or similar facility are not simulated. However, a private person's activity schedule may include delivering a parcel to a post office or similar facility. Therefore, only the initial part of a transport chain for sending parcels by private individuals is implicitly modeled. Additionally, a private individual's activity schedule might involve directly delivering a parcel to another private individual without utilizing the services of a CEPSP. Businesses can send parcels outside the study area using CEPSPs or to other private individuals and businesses within the study area. This necessitates

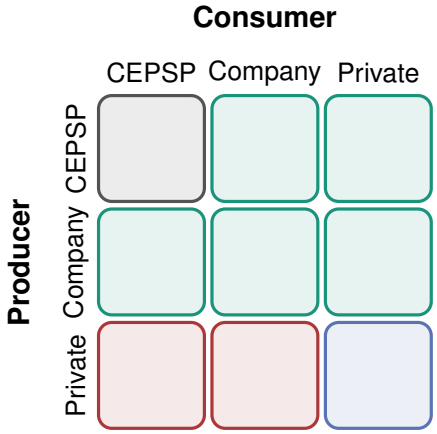


Figure 2.8: Supported parcel producer-consumer relations in logiTopp: green - supported, red - not supported, gray - irrelevant, blue - implicitly modeled (adapted from [9]).

the company to operate its logistic network. Although this mechanism is theoretically supported, it hasn't been studied yet due to insufficient data available to generate the corresponding demand. Another distinguishing feature of logiTopp is the representation of complex transport chains. Parcels can be temporarily stored and reloaded several times at depots and transported on multi-modal transport chains. Which allows the representation of complex logistic networks.

In terms of structure, logiTopp extends the existing long and short-term modules. Figure 2.9 and 2.10 give an overview of logiTopp's long, respectively, short-term module. A high-level overview of the module's processing steps is shown in the first column. The second column describes the corresponding program flow in a flowchart-like format and partially distinguishes between processing steps relevant to business or private parcels. LogiTopp is an extendable framework and provides various extension points which are shown in the third column and linked to the point of the program flow where they are applied. Grayed-out extension points are lazy and thus evaluated in the short-term module at runtime.

In the long-term module, illustrated in Figure 2.9, the initial population of private and business agents is created. The preferred packstation for each private agent is also determined. Subsequently, the module generates the parcels consumed and produced by both business and private agents throughout the entire simulation period. Each parcel's delivery date, delivery location, and size are then determined. However, the responsible carrier and, consequently, the origin or destination depot are not determined at this stage. Instead, they are calculated lazily, taking into account market share distributions, contractual relations, and the current workload of the carriers. Finally, the module derives the contractual relations between business agents and carriers.

The short-term module (see Figure 2.10) encompasses a dynamic simulation. Within

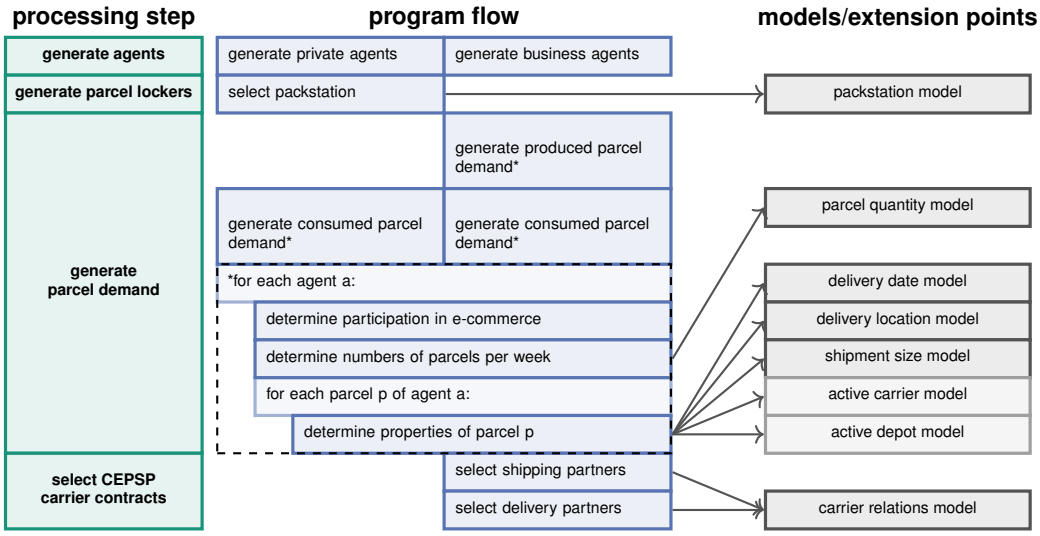


Figure 2.9: Long-term module of logiTopp - processing steps, program flow with therein used extension points and models (adapted from [30]).

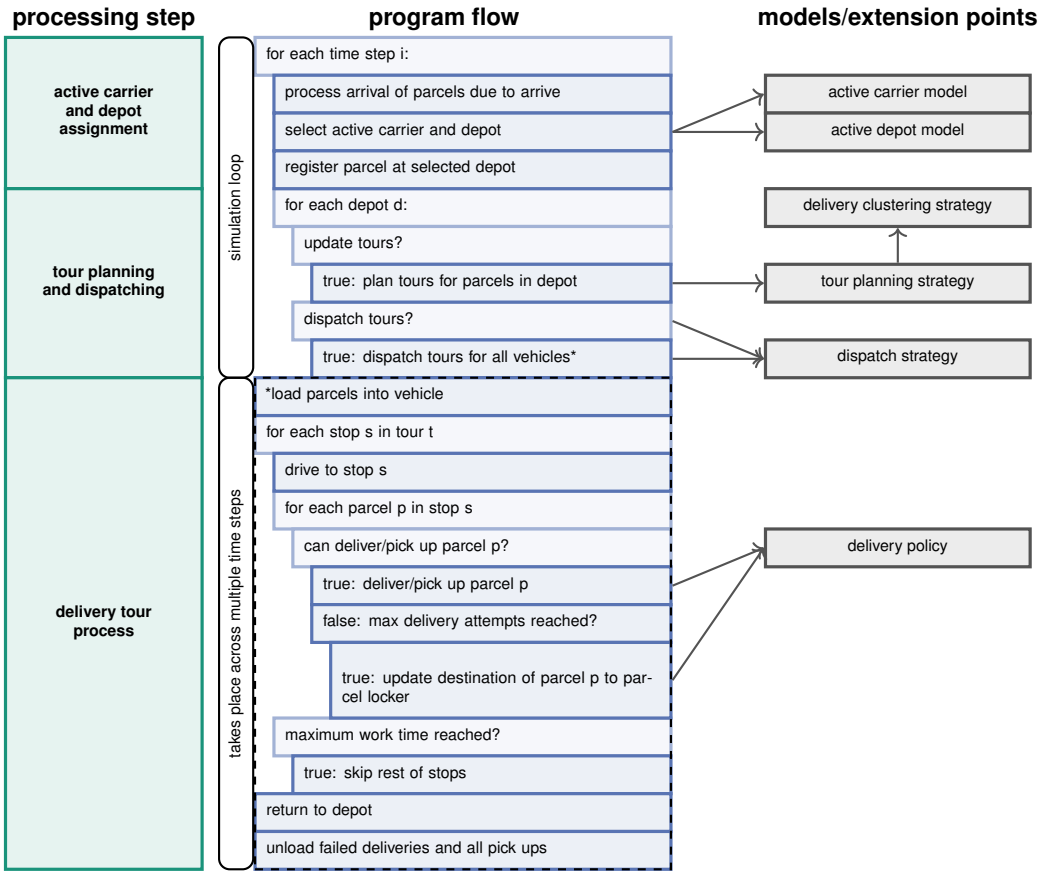


Figure 2.10: Short-term module of logiTopp - processing steps, program flow with therein used extension points and models (adapted from [30]).

each time step of the simulation, some parcels are potentially inserted into the simulation. This is where the responsible carrier, and consequently, the origin or destination depot for the parcels, is determined. Then, the responsible depot gets informed that the parcel has arrived or needs to be picked up. The depots operate a fleet of delivery vehicles and periodically plan and dispatch tours to handle the new parcels. A tour contains several stops where parcels have to be picked up or delivered. The tour is then carried out over multiple time steps of the simulation. At the beginning, the parcels are loaded into the vehicle. Then, the planned stops are approached sequentially. It is decided for each parcel that has to be picked up or delivered at the current stop if it can be picked up or delivered at the current time step. This takes into account, for example, whether the recipient, a member of the recipient's household, or at least a neighbor is at home. If a delivery is not possible, the parcel remains in the vehicle and gets returned to the depot or, after some missed attempts, gets redirected to a parcel locker. The picked-up and returned parcels are unloaded back at the depot, and it's the depot's responsibility to plan further tours to deliver the parcel or transfer it to another depot. It is also possible that a tour exceeds the driver's maximum working time and, therefore, has to be terminated prematurely.

MATSim-Freight

MATSim [2, 7] is an activity-based, extendable, multi-agent simulation framework. It was initially developed for passenger transport simulation but has been expanded in many ways. The freight contribution [53, 54, 61, 62] added carrier agents into the simulation. Joubert et al. [10] extended this further by modeling the behavior of freight receivers. MATSim and its contributions are publicly available on GitHub [2]. This section begins with an introduction to MATSim and its basic principles, followed by a brief introduction to the extensions for freight transportation.

MATSim is an agent-based transport simulation. That means every member of the population is an agent with a plan, which is a schedule of activities the agent has to perform during the one-day simulation period. The simulation process is shown in Figure 2.11 and explained briefly in the following. The first step generates the *initial demand*, i.e., the agent's initial plan,. This can be done, for instance, using empirical data. The

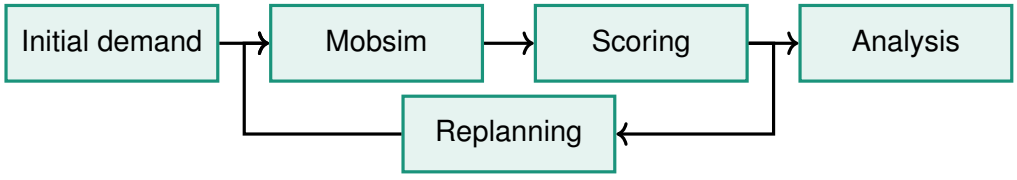


Figure 2.11: MATSim process (adapted from [7]).

Mobsim (mobility simulation) step is an execution of the activity schedule in a usually queue-based traffic flow model. Next, the performance of each plan is determined in the *Scoring* step using a scoring function. In the *Replanning* step, a fraction of the agents are selected, and their current plan is cloned and replanned by mutation. The mutation is achieved by applying replanning modules that can change a single-choice dimension of an existing plan, e.g., destination choice or departure times. An agent has a limited memory of plans with their associated scores. After the *Replanning* step, the agent selects one of the available plans. If the memory is full, the plan with the lowest score gets removed. This process is repeated by a fixed number of iterations. Afterward, the *Analysis* module can be used to display and analyze the results and intermediate results of the simulation in various forms. This process follows the co-evolutionary principle [48] where agents repeatably optimize their activity schedule whilst competing with other agents for space and time resources in the transport network. The result of this iterative process is a stochastic user equilibrium. MATSim is designed for large-scale scenarios and provides support for parallel computation. The modular design ensures that MATSim can be flexibly customized and easily extended.

The freight contribution [53, 54, 61, 62], also referred to as MATSim-Freight or Freight Transport Lab, adds another type of agents into the simulation to enable more realistic modeling of freight transport behavior. These carrier agents represent logistic companies with a fleet, depots, and orders. Orders specify the deliveries to be made by the carrier with the type and quantity of delivered goods and time windows for pickup and deliveries. A carrier agent's plan is a tour schedule for the fleet vehicles, specifying pickup and delivery times and a route. In the *Mobsim* step, the carrier vehicles are inserted into the existing traffic simulation. The *Scoring* step evaluates the commercial success of carrier plans by assessing the fulfillment of customer requirements, such as adherence to time windows, and the costs of executing the plan, e.g., personnel and transportation costs. Possible mutations applied during the *Replanning* step involve rerouting vehicles, switching deliveries between vehicles, or adding or removing a vehicle from the schedule. For the creation of the initial plans and potentially in the *Replanning* step, the external library jsprit [1] is used, which is an implementation of the algorithm described by Schrimpf et al. [52] with strategies inspired by the work of Pisinger and Ropke [46].

A further addition to this model has been proposed by Joubert et al. [10], also referred to as the freight receiver contribution. Instead of modeling the behavior of freight receivers only statically through their placed orders, the freight receivers become dynamic agents themselves and can decide on the size, frequency, and timing of their orders. This is referred to as the plan of a receiver. The degree of freedom of a receiver's plan is constrained by the storage capacity and actual demand of a receiver. The selected plan of a receiver is then translated and inserted as orders into the existing infrastructure

of the freight contribution. During the *Scoring* step, the commercial success of a plan is evaluated. This score is determined as a function of the different carriers' selected plan delivery costs. The mutation of receiver plans during the *Replanning* step involves the adjustment of delivery time windows, unloading times, or delivery frequency. This addition adds a new layer of interaction between receivers and carriers. However, the chosen receiver modeling mainly applies to large industrial receivers or, at least, businesses in general, but not private receivers. Since it takes some iterations for a carrier to react and optimize for a new receiver's plan, the original simulation process has to be adapted. This is shown in Figure 2.12. Basically, instead of changing its plan every iteration, a receiver is only allowed to change its plan during a replanning iteration, e.g., every 10th iteration. After such a change, the new plan is inserted as orders into the existing infrastructure of the freight contribution, and new initial plans for the carrier are generated using the jsprit [1] library.

Other Models

In the realm of freight transport modeling, a variety of models have been developed, each offering unique insights and approaches. In this section, we provide a brief overview of some notable models, focusing particularly on microscopic, disaggregated, or agent-based methodologies, with a specific emphasis on their relevance to the CEP sector. We refer the reader to the previously mentioned overview and fundamental works for a broader and more complete overview.

One pioneering model in this domain is the GoodTrip model, conceived by Boerkamps and van Binsbergen [13]. GoodTrip introduced a conceptual framework centered on goods, transport services, and traffic markets. It employs supply chain formulations to describe goods flows derived from supply and demand, which are then translated into traffic flows. The work of Nuzzolo and Comi [41] follows a similar approach. Another well-known model is MASS-GT, developed by de Bok and Tavasszy [19], which uses

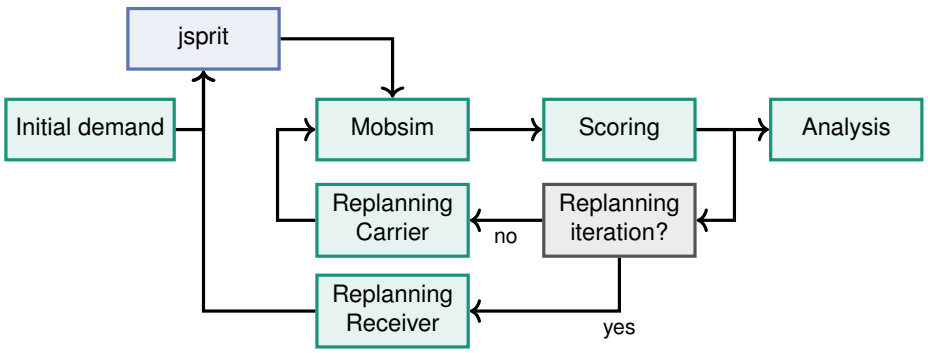


Figure 2.12: Modified process of the MATSim freight receivers contribution (adapted from [10]).

an extensive commodity flow database.

Fischer et al. [23] proposed a model that integrates logistic chain-based and tour-based approaches into the modeling framework. Their model allows the representation of concepts relevant to the CEP market, such as pickup and delivery at households and distribution centers.

In their work on FREMIS, Cavalante and Roorda [16] presented a detailed conceptual framework for the freight transportation domain. Although not implemented, the framework offers explicit and detailed formulations of contractual relations between various domain entities, which are gradually refined and transformed into actual demand and tours performed. This allows, for example, the realization of subcontractors or companies that only act as brokers for transport services.

SimMobility [3] and its freight extension, SimMobility Freight [5, 51], stand out as a detailed and highly developed model, with a focus on urban settings. Notably, an interesting dimension they consider is parking choices for delivery vehicles, addressing spatial constraints in urban areas.

Examples of models explicitly incorporating the CEP sector are the models developed by Dalla Chiara et al. [18] and Llorca and Moeckel [34]. The latter incorporates a distinction between origin-to-destination truck trips and transport chains via distribution centers. Shipments using complex transport chains are then planned in a multi-stage process explicitly separating long-distance truck trips and trips for urban deliveries with various transport modes possible.

2.4 Related Work

In this section, we offer an overview of related work. While this foundation chapter has already provided references to scientific works related to and used in this study, this section aims to fill any remaining gaps. Therefore, this section focuses on presenting works with similar goals and explores the foundations of possible alternative approaches to addressing the investigated problem.

Briem et al. [14] developed in their work a coupling of mobiTopp and MATSim, with the goal of integrating the mesoscopic traffic simulation of MATSim in mobiTopp. The coupling allows feedback loops between the short-term module of mobiTopp and the simulation and optimization of route choices within MATSim. A similar approach is explored by Bekhor et al. [11] by again integrating a dynamic traffic assignment framework into an activity-based model. Further, many, primarily commercial transport models, such as Aimsun Next [4], offer various import and export functionalities to support data exchange with other models. In their work, Erdelyi et al. [21] explore approaches to

multi-level model coupling of mesoscopic and microscopic traffic simulations. However, all these approaches are limited to coupling a single pair of models, and the author is unaware of any work that explicitly attempts to couple a variety of traffic models in an integrated approach. Furthermore, no work is known that explicitly couples freight transport models.

This work realizes the coupling through model transformations and a central common metamodel. A similar but yet different approach that could potentially employed is the idea of a virtual single underlying metamodel (V-SUM) used in the VITRUVIUS approach [27]. Instead of creating a metamodel comprising all aspects of a system, a V-SUM is composed of several existing metamodels, which are then continuously kept consistent by using consistency-preserving operations derived from changes to one of the models. One potential benefit of this approach would be supporting more iterative and fast processes, as small changes to one model can be propagated very efficiently to the other integrated models.

Another central property of this work is the explicit formulation of a domain model for freight transport. Using domain models is a widespread technique, especially in computer science. Many of the previously presented freight transport models incorporate a conceptual framework for this domain. However, most of these frameworks focus on formulating relations and processes between the domain's stakeholders and do not explicitly handle the data relevant to data exchange. There does not appear to be a universally valid domain model or exchange format for data and problem definitions from the freight transport and CEP domains. Other domains have such widely accepted standards. An example is the railML format [39], used in the railway industry and research.

3 Freight Transport Metamodels

In this chapter, we develop, analyze, and compare metamodels that represent the structure of logiTopp and MATSim-Freight, as well as selected concepts of other metamodels. This serves as the basis for the development of a coupling concept (see Chapter 4), a common metamodel, and the definition of model transformations (see Chapter 5) in subsequent chapters. We begin by describing the methodology used for both the development and analysis of the two metamodels. Following this methodology, we present and analyze the metamodels of logiTopp (Section 3.1) and MATSim-Freight (Section 3.2). Additionally, in Section 3.3, we briefly describe selected concepts of metamodels used in further freight transport models. This chapter finishes with a comparison of the presented metamodels in Section 3.4.

The presented metamodels of logiTopp and MATSim-Freight were reverse-engineered from the available source code on GitHub [2, 30]. Furthermore, the existing work on the models provided additional information. In Section 3.3, we solely relied on work presenting the discussed models. The metamodels are instances of the Ecore meta-metamodel of the EMF [57]. Only the elements related to freight transport, along with some foundational concepts from the domain models, were included in the metamodels. The presented freight transport models use partially dynamic simulation so that its domain model elements partially describe the state of the simulation. During the creation of the metamodels, we intentionally excluded elements representing a state whenever possible. Instead, our focus remained solely on the static data model. This choice simplified the resulting metamodels, enhancing their comprehension and manageability. Moreover, the primary aim of this work, model coupling, necessitates concentrating solely on static data. A coupling of potentially inhomogeneous models during a simulation, and thus the transfer of stateful data, appears unrealistic or at least beyond the scope of this work. Furthermore, we simplified the resultant metamodels by eliminating redundancies, renaming elements, and abstracting from some implementation-related structures.

We describe the abstract syntax of the metamodels through class diagrams, employing the notation derived from EMF's Ecore diagram editor [57]. An example that introduces parts of this notation is given in Section 2.1.3. The static and dynamic semantics will be explained using natural language descriptions. However, a complete description of

the static semantics has been omitted. This would go beyond the scope and would not contribute to a deeper understanding of the concepts, as a large part of the constraints can be derived simply from the abstract syntax.

The presentation, analysis, and comparison of the metamodels in this chapter are based on the concept of viewpoints. A viewpoint is a conceptual perspective that serves a specific concern [24]. In this work, a set of six viewpoints is utilized and briefly introduced as follows:

- Network Viewpoint: The network viewpoint focuses on the representation and structure of the transportation network that can be used by private persons, businesses, and transport service providers. It includes elements such as nodes, links, transport modes, and their interconnections. Furthermore, the partitioning and structuring of the area is part of the network viewpoint.
- Population Viewpoint: The population viewpoint deals with entities involved in creating demand for freight transport, emphasizing the description and characteristics of individuals and organizations within the model. It aims to capture the demographic and economic aspects of freight transport demand and includes the definition of persons, households, businesses, and other relevant entities, along with their attributes, roles, and relationships.
- Logistic Demand Viewpoint: The logistic demand viewpoint concentrates on the representation of freight transport demand. It encompasses the definition of shipment requests, delivery orders, or service requests. This viewpoint addresses the spatial and temporal aspects of demand, considering factors such as delivery locations, quantities, and time windows.
- Transport Infrastructure Viewpoint: The transport infrastructure viewpoint focuses on the physical and organizational aspects of the infrastructure and companies that handle freight transport. This includes depots, vehicles, and other facilities that play a role in the transportation process as well as organizational aspects like the representation of transport chains. It addresses attributes, capacities, and connections of these infrastructure and organizational elements, providing a comprehensive understanding of the logistical network.
- Logistic Solution Viewpoint: The logistic solution viewpoint outlines how the logistical network is utilized to meet the logistic demand. It includes elements related to the planning and execution of transportation activities, such as carrier assignment, tour planning, routing decisions, and scheduling.
- Results Viewpoint: The results viewpoint focuses on the output and outcomes derived from the execution of the freight transport model. It includes metrics, reports, and data generated during or after the simulation. This viewpoint enables the analysis and evaluation of the model's results.

For each presented metamodel and each viewpoint, we created a view type. A view type defines a subset of the original metamodel that pertains to the concern addressed by the viewpoint. Consequently, a view type exclusively encompasses metaclasses, attributes, and relations related to the specific concern. This implies that a metaclass might potentially appear in multiple view types, each featuring different attributes. Further, we omitted the explicit presentation and description of opposite references during the presentation of the metamodels in order not to overload the diagrams and descriptions.

3.1 LogiTopp

Before presenting the single view types of the logiTopp metamodel, we discuss some cross-cutting concerns used in the whole metamodel.

LogiTopp has a typical simulation period of one week, and the smallest unit of time is a second. In the current implementation, it is convention that the simulation starts at midnight on Monday morning. Two metaclasses are provided to express times. *Time* represents a point in time within the simulation period and provides the attributes *dayOfWeek*, *hour*, *minute*, and *second*, collectively representing the day of the week, hour, minute, and second components in the format *dd hh:mm:ss*. A special case is that it is also possible to express a not-yet-defined point in time that lies in the future. *RelativeTime* is used to express durations in a similar format than *Time*. With *RelativeTime*, it is also possible to express infinite or undefined durations.

LogiTopp is not fixed on a specific coordinate system, but the use of cartesian coordinates is recommended. The selected unit of the coordinate system is then used as the base unit for other attributes specifying distances and speeds. A typical selected unit is meters. LogiTopp uses only volumes to specify parcel dimensions and capacities of logistic facilities. Volumes are always given in cm^3.

LogiTopp uses the *id* attribute as a unique identifier for instances of a class, enabling distinct identification and referencing of entities within the model. In logiTopp *id* attributes are represented as integer values.

3.1.1 Network View Type

The network model of logiTopp comprises a simple directed graph. A graph is composed of *edges* and *nodes*. Edges are directed, maintaining a fixed traversal direction defined by their start (*from*) and endpoint (*to*). In cases where a link between two nodes permits travel in both directions, two edges exist, each representing a direction. These

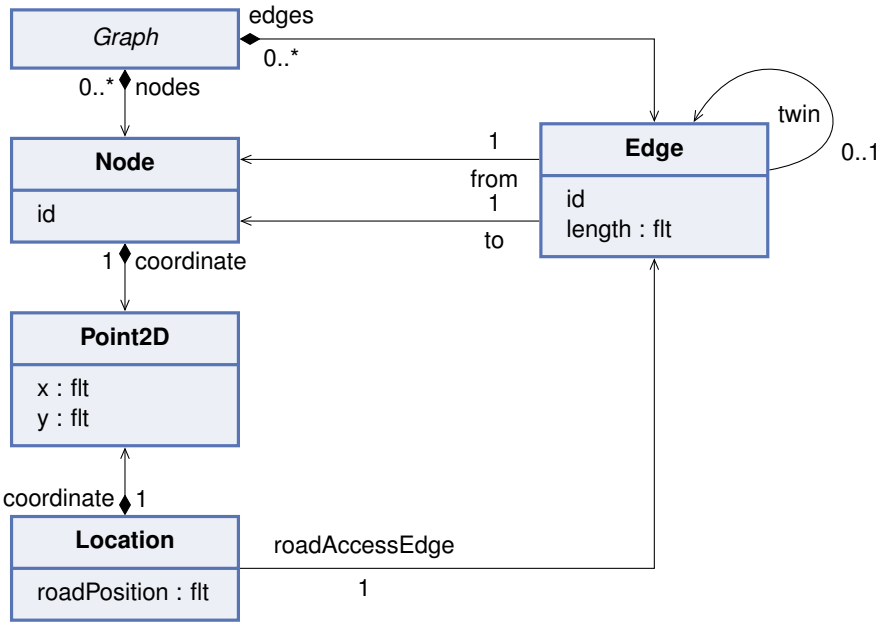


Figure 3.1: LogiTopp Metamodel: Network View Type - Network graph.

edges reference each other's opposite edge (*twin*). Nodes are positioned within the chosen coordinate system using *x* and *y* coordinates (*Point2D*). Additionally, edges possess a *length* attribute, referring to the unit utilized in the coordinate system. This feature enables the modeling of curved edges, allowing for lengths different from the Euclidean distance. LogiTopp uses the concept of *Locations*, which encompasses not only coordinates but also specifies the point at which to access the network from that location. This is accomplished by referencing the edge through which the network is accessed (*roadAccessEdge*), along with its relative position on that edge (*roadPosition*), modeled by a value between zero and one. Figure 3.1 shows the network graph of logiTopp's metamodel.

The *RoadNetwork* used in logiTopp extends the network graph by incorporating the concept of *zones*, commonly used in transport modeling. The study area is subdivided into a set of zones aimed at grouping geographic areas with similar characteristics. These zones provide aggregated data that can be utilized during simulation. However, logiTopp only models the geographical extent of the zones and currently does not hold any aggregated data. The corresponding metamodel elements are shown in Figure 3.2. Subsequently, we briefly describe these elements:

- A *RoadNetwork* is a *Graph* with a set of *zones*.
- Each *Zone* has an *id* and a *name*. Its geographical extent is described by a *ZoneArea* (*totalArea*). Also, the *centroid* of a zone is modeled by a *Location*.
- A *ZoneArea* describes the geographical extent of a zone by a set of non-intersecting *Polygons* (*borders*). Additionally, the size of the *ZoneArea* is de-

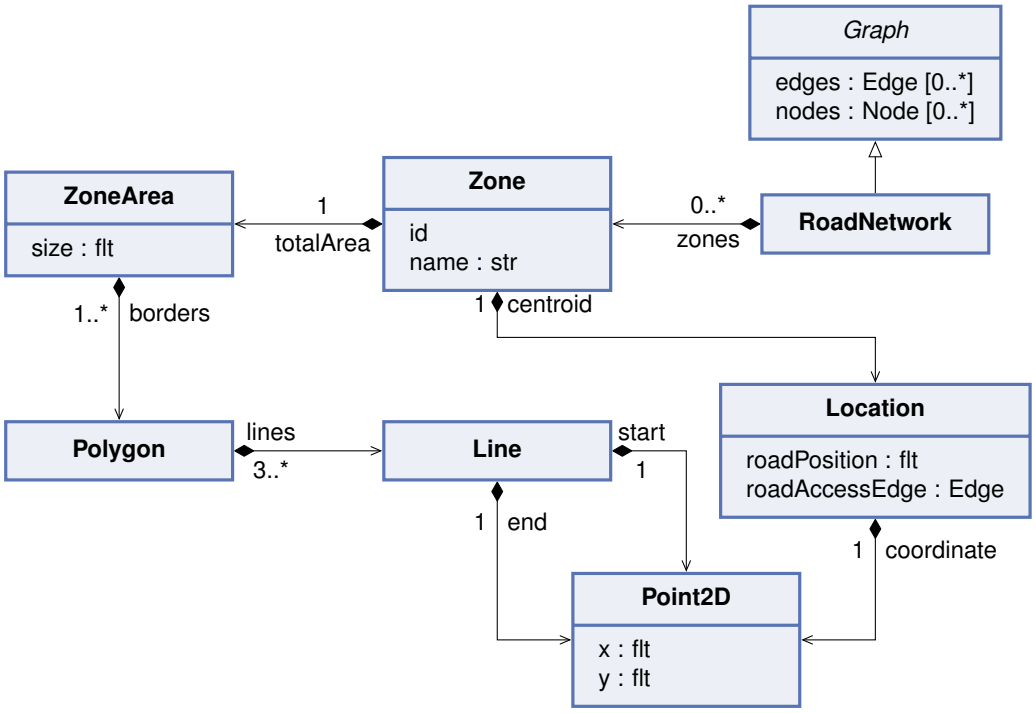


Figure 3.2: LogiTopp metamodel: Network View Type - Road network with zones.

scribed by the *size* attribute.

- *Polygons* are described by their outline. Thus, they consist of at least three *Lines*. The endpoint of a line has to be at the same location as the startpoint of the subsequent line. Additionally, the endpoint of the last line has to be at the same location as the startpoint of the first line.
- A *Line* has a startpoint (*start*) and an endpoint (*end*).

Information about travel times is not directly included in the network graph. Instead, they are represented by a travel time matrix, that represents the time it takes to travel between different origin-destination pairs (nodes) within the network.

The *RoadNetwork* serves as the root element for the network-related elements.

3.1.2 Population View Type

The population viewpoint consists of the part of the metamodel that describes entities responsible for generating demand for the transport network. LogiTopp has two entities that receive or send packages: businesses and private persons.

Figure 3.3 shows the parts of the logiTopp metamodel representing businesses. Subsequently, we briefly describe these elements:

- The enum *Weekday* lists the days of the week.

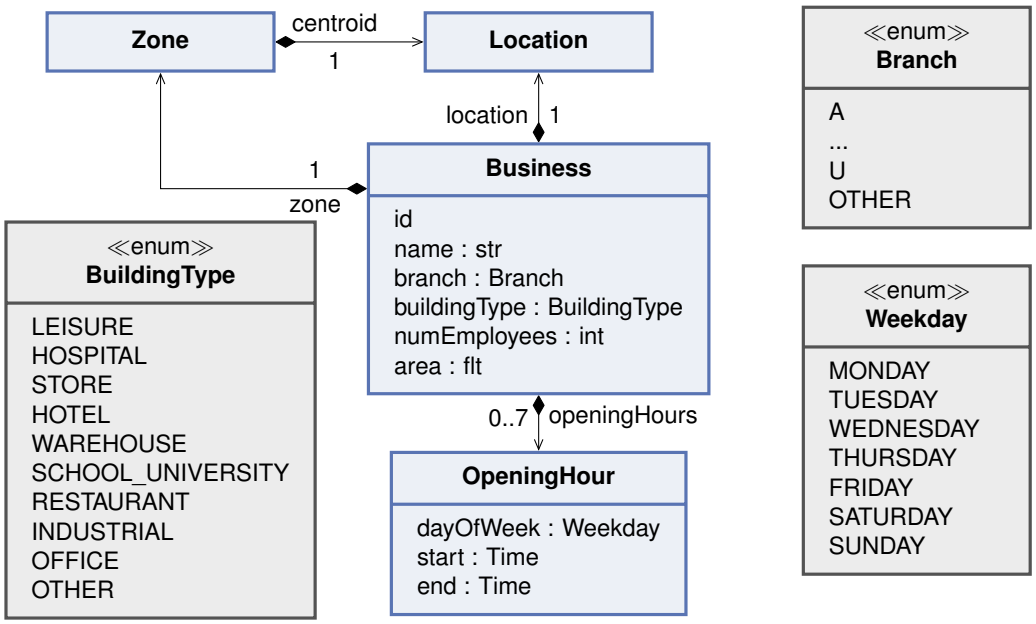


Figure 3.3: LogiTopp metamodel: Population View Type - Business.

- *Branch* models the branch categories according to the standardized NACE classification [22].
- *BuildingType* is another higher-level sector-based classification of branch or building categories. The implementation of logiTopp also provides an $n:n$ mapping between *BuildingType* and *Branch*.
- An *OpeningHour* describes the opening hours of a business. It defines the *dayOfWeek*, as well as a time interval during that day when the business is open. The time interval is defined by its *start* and *end* time.
- A *Business* has an *id* and a name. It is geographically localized by a *Location* and a *Zone* in which it is situated. The type of business is described by the attributes *branch* and *buildingType*. Additionally, the number of employees (*numEmployees*) and the commercial *area* of the business are given. In terms of *openingHours*, just one *OpeningHour* per day of the week is allowed. Which leads to a maximum of 7 contained *openingHours*.

Figure 3.4 shows the parts of the logiTopp metamodel representing private persons that live together in households. Subsequently, we briefly describe these elements:

- The enum *Employment* is used to describe the employment status of a person. It encompasses a wide range of employment and non-employment statuses, including various work-related states, educational pursuits, household roles, and stages of life.
- The enum *Gender* lists possible genders.
- *Persons* are elementary agents of logiTopp respectively mobiTopp and there-

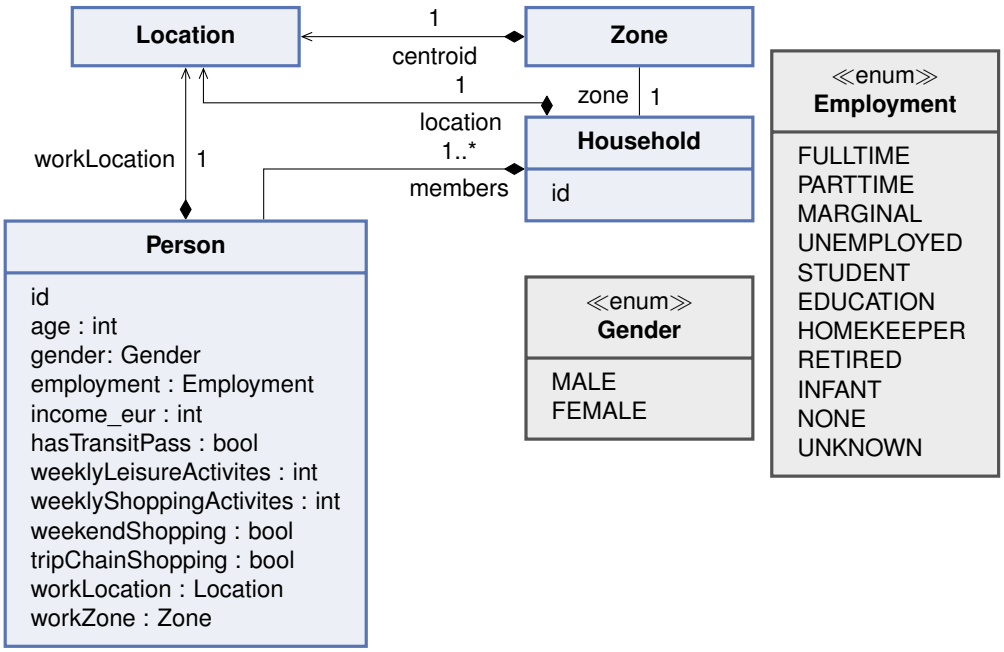


Figure 3.4: LogiTopp metamodel: Population View Type - Persons and households.

fore have many attributes. This includes an extensive activity schedule that is initialized by mobiTopp's long-term module. LogiTopp allows the use of custom demand generation models that can make use of all of these attributes. To reduce complexity, we included only attributes used by current research projects. Some of these attributes are derived from the generated activity schedule. A *Person* as an *age* and a *gender*. Also, the *employment* status as well as the economic status (*income_eur*) is modeled. The attribute *hasTransitPass* describes whether a *Person* owns a transit pass or not. The shopping behavior of a *Person* is described by the number of weekly shopping (*weeklyShoppingActivities*) and leisure (*weeklyLeisureActivities*) activities, whether a person also carries out shopping activities at weekends (*weekendShopping*), and whether shopping activities are integrated into other trips or carried out separately (*tripChainShopping*). Further, if a *Person* is employed, its workplace is located by a *workLocation* and *workZone*.

- A *Person* is a member of exactly one *Household*. A *Household* can have multiple members. It is also geographically localized by a *location* and a *zone* in which it is situated.

Further, a not shown root element (*Population*) exists that serves as the root element for all *Businesses* and *Households*.

3.1.3 Logistic Demand View Type

LogiTopp's logistic demand consists of parcels that need to be delivered or picked up. Section 2.3.2, respectively Figure 2.8, describes supported producer-consumer relations supported in logiTopp. Note that parcels sent into or leaving the study area enter or exit the study area through a CEPSP. It should also be noted that parcels sent within the study area and handled by a CEPSP result in two separate parcels: one from the origin to the CEPSP and the second from the CEPSP to the destination.

Figure 3.5 shows the concept of parcel producers and consumers of the logiTopp metamodel. This can mainly be seen as a role-based concept. A *ParcelConsumer* is an entity capable of receiving parcels, while a *ParcelProducer* can send parcels. Common characteristics of both are summarized in the superclass *ParcelAgent*. A Person only acts as *ParcelConsumer*, while *DistributionCenters* and *Businesses* act as both: *ParcelConsumer* and *ParcelProducer*. Note, that not the CEPSP direct and instead a *DistributionCenter* of the CEPSP acts as *ParcelAgent*. This has the benefit of providing a concrete location for the origin or destination of the parcel.

The modeling of a parcel is shown in Figure 3.6. Subsequently, we briefly describe the shown elements:

- The *ParcelDestinationType* specifies the actual destination of a parcel sent to a *Person*, which could be a packstation, the recipient's household, or their workplace.
- The *ShipmentSize* describes the size of a *Parcel*. The logiTopp implementation also provides a mapping that assigns a volume range to each category.
- A *Parcel* is the core unit of demand in logiTopp. Each parcel has an *id*, a *producer*, and a *consumer*. Additionally, the precise delivery location for the package is specified by a *zone* and a *location*. This specification is necessary because the delivery location does not always correspond to the consumer's location,

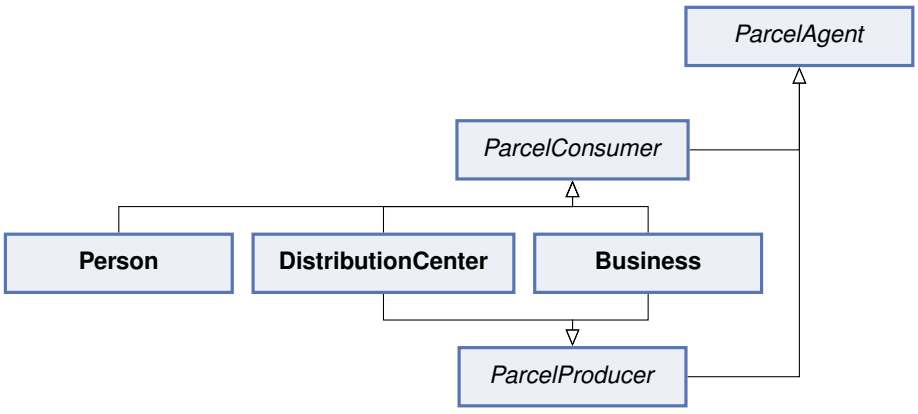


Figure 3.5: LogiTopp metamodel: Demand View Type - Parcel consumer and producer.

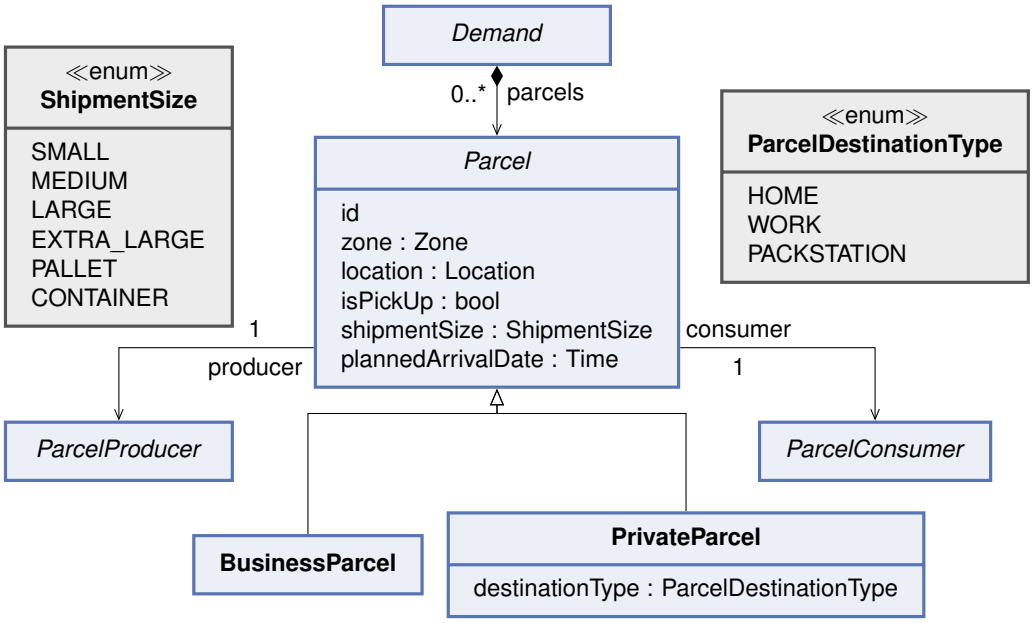


Figure 3.6: LogiTopp metamodel: Demand View Type - Parcels.

for instance, when delivering to a packstation. The attribute *isPickUp* denotes whether a CEPSP needs to pick up the parcel. In this case, the pickup location can be derived from the consumer's location. Moreover, it includes information on the parcel's volume (*shipmentSize*) and the time the parcel arrives at its initial location (*plannedArrivalDate*).

- *PrivateParcels* are parcels received by a *Person* and enter the study area via a CEPSP. Therefore, the *producer* and *consumer* associations are restricted to *DistributionCenters* and *Persons*, respectively. Additionally, a *PrivateParcel* has a *destinationType* that describes the actual destination.
- A *BusinessParcel* is a parcel sent or received by a *Business*. The *producer* association is restricted to *Businesses* and *DistributionCenters*. Every *ParcelConsumer* is a potential *consumer*. However, either the *producer* or the *consumer* must be a *Business*.
- All *Parcels* are contained in a single root element called *Demand*.

3.1.4 Transport Infrastructure View Type

The transport infrastructure viewpoint comprises the parts of the metamodel that describe the logistics network. The central unit for providing logistic services in logiTopp is the *CEPServiceProvider*, characterized by its *name*. LogiTopp explicitly models a contractual relationship between *Businesses* and *CEPServiceProviders*, depicted in Figure 3.7. The contractual relationships of a *Business* are differentiated into partners for sending (*pickUpPartners*) and receiving (*deliveryPartners*) parcels.

Figure 3.7: LogiTopp metamodel: Transport Infrastructure View Type - Contractual relations between businesses and CEPSPs.

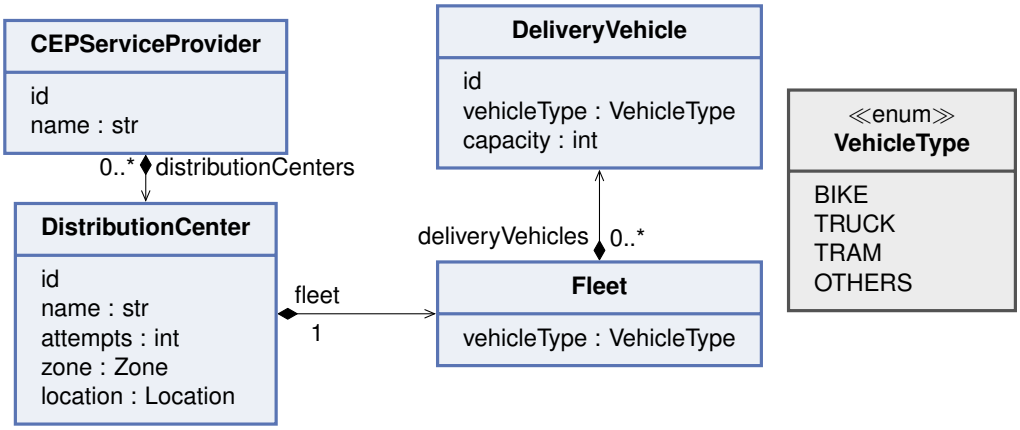


Figure 3.8: LogiTopp metamodel: Transport Infrastructure View Type - Resources of CEPSPs.

Figure 3.8 shows the resources operated by a *CEPServiceProvider*. This includes *DistributionCenters* and a fleet of *DeliveryVehicles*. Subsequently, we briefly describe these elements:

- The enum *VehicleType* is used to describe the type of a vehicle.
- A *DeliveryVehicle* has a *vehicleType* and a *capacity*. The *capacity* corresponds to the maximum volume of parcels that the vehicle can handle.
- A *Fleet* consists of a set of *DeliveryVehicles*. The model is restricted to homogeneous fleets concerning the *VehicleType* of its vehicles, specified in the attribute *vehicleType*.
- *DistributionCenters* contain exactly one *Fleet* and are geographically localized by a *Location* and a *Zone* in which they are situated. The attribute *attempts* defines how often the vehicles of a *DistributionCenter* attempt to deliver a parcel to the designated delivery location before it gets redirected to a pack station.
- Every *CEPServiceProvider* operates and contains a set of *DistributionCenters*.

LogiTopp allows the modeling of complex logistic networks. A *DistributionCenter* has a designated area of responsibility (*ServiceArea*) where it can deliver parcels. Additionally, parcels can be transferred between different distribution centers, which forms a logistic network that implicitly models transport chains. Subsequently, we briefly describe these elements:

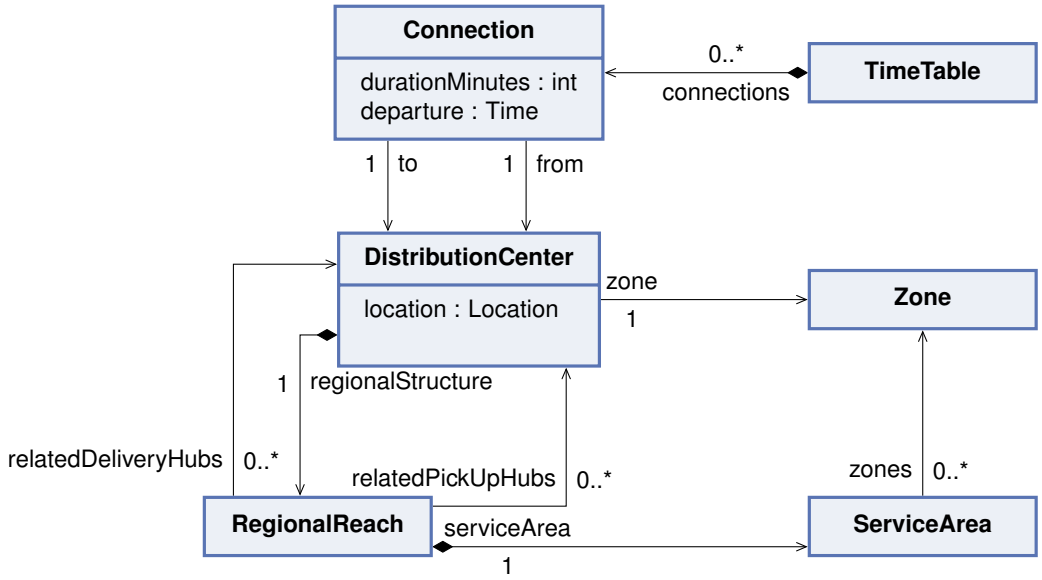


Figure 3.9: LogiTopp metamodel: Transport Infrastructure View Type - Logistic network and regional distribution.

- *ServiceAreas* delineate specific regions within the study area where distribution centers can both deliver and pick up parcels. These areas are defined by a set of *zones*.
- The *RegionalReach* serves as the fundamental element in describing the transport network. It defines the *serviceArea* of a *DistributionCenter* and its connections to other *DistributionCenters*. These connections are represented by two sets: *relatedDeliveryHubs* for parcel delivery and *relatedPickUpHubs* for parcel pickups. This structure allows a *DistributionCenter* to plan a direct tour to a parcel's destination within its *serviceArea* or transfer the parcel to a *relatedDeliveryHub* for further delivery, by planning a tour to this *DistributionCenter*. In the case of a parcel pickup, the behavior is analogous. Each *DistributionCenter* has exactly one *RegionalReach*.
- A *Connection* describes a tram unidirectional journey between two DistributionCenters (*from*, *to*), that starts at a defined *departure* time and has a specific duration (*durationMinutes*), which is stated in minutes.
- *DistributionCenters* that operate trams must adhere to the *TimeTable*. The *TimeTable* is a global entity that contains all tram journeys that potentially can be used to transport parcels within the study area (*connections*). There is no distinction between different weekdays, so the time *TimeTable* describes all possible connections within one day. Every planned tour operated by a tram must realize and comply with the constraints of a *Connection*. It is also not allowed to plan and execute multiple tours realizing the same *Connection*.

Note that the *relatedDeliverHubs* and *relatedPickUpHubs* of the *RegionalReach* are

not constrained to facilities of the same CEPSP. This flexibility allows multiple CEPSPs to participate in the handling of a parcel. Shared resources, such as trams, can be modeled as a separate CEPSP to which all other CEPSPs can transfer parcels. However, the representation of a *RegionalReach* cannot distinguish the original CEPSP that handled a parcel, and consequently, there is no restriction on further transfers to other hubs based on this information.

The relationships defined by the *RegionalReach* construct two distinct directed graphs for the delivery and pickup of parcels. In these graphs, *DistributionCenters* and their associated *ServiceAreas* (when they contain at least one zone) serve as nodes. The edges represent connections through *relatedDeliverHubs* and *relatedPickUpHubs*, as well as the relationship between *DistributionCenters* and their *ServiceAreas*. Both *ServiceAreas* and the *DistributionCenters* through which parcels enter or exit the study area act as sources and sinks, depending on the direction (delivery or pickup). All paths between all pairs of sources and skinks represent potential transport chains. It is a metamodel constraint that the resulting graph must be acyclic.

The logiTopp implementation also has an explicit representation of transport chains. This is redundant to the previously presented concept of related hubs and, therefore, has been omitted. Indeed, logiTopp itself is capable of generating transport chains from the information about related hubs.

Furthermore, there is a root element (*TransportInfrastructre*) that contains all *CEPServiceProder* instances and a single *TimeTable* instance.

3.1.5 Logistic Solution View Type

During the simulation, a *DistributionCenter* dynamically plans and executes tours to pick up, transfer, and deliver the parcels it currently contains or is responsible for. The metamodel elements used to describe these tours are shown in Figure 3.10. Subsequently, we briefly describe these elements:

- Each *DistributionCenter* has a *DepotStorage*, primarily used to track the current parcels for which a *DistributionCenter* is responsible. It also maintains records of the currently *plannedTours*, which is the sole relevant attribute in the logistic solution view.
- A *PlannedDeliveryTour* has an *id* and describes a planned tour, represented by an ordered list of *stops*. It is designed for a specific *vehicleType* and includes the start time (*plannedAt*) and the *plannedDuration* of the tour. Although there is no concept of routing through the network graph, the tour planning algorithms rely on the network graph and given travel times to ensure consistent and realistic planned travel times.

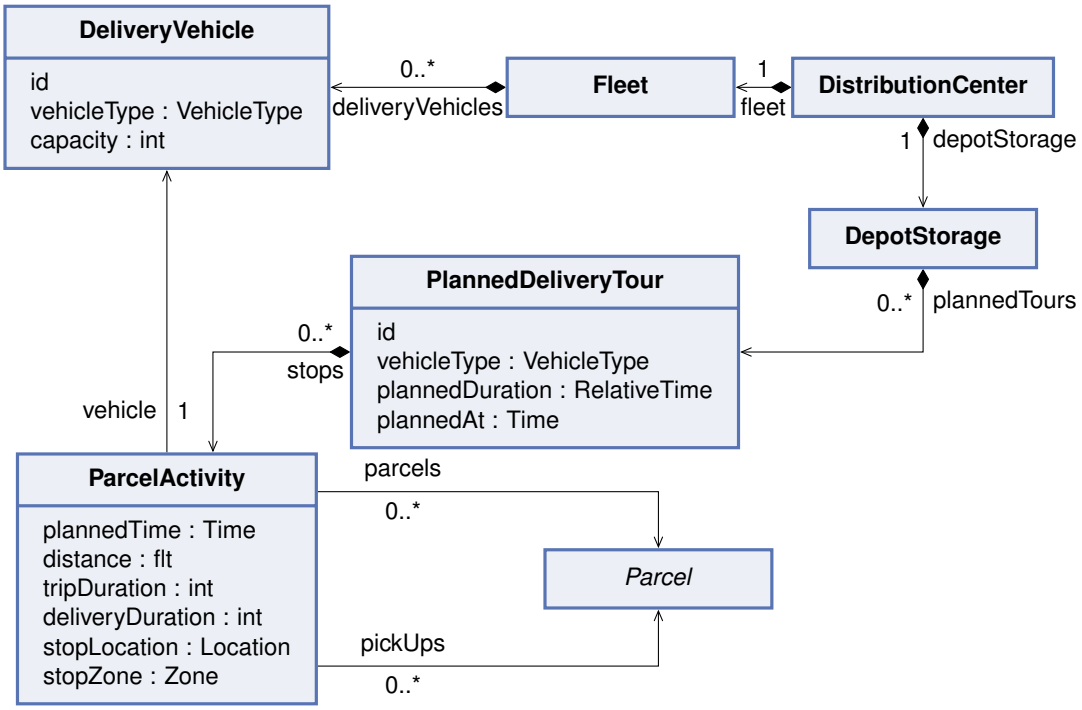


Figure 3.10: LogiTopp metamodel: Logistic Solution View Type.

- The stops on a tour are denoted by *ParcelActivities*. At each stop, a set of *Parcels* can be delivered (*parcels*) and picked up (*pickUps*). This activity takes time, described by a *deliverDuration* in minutes. Each stop is located by a *stopLocation* and *stopZone*. Further attributes include the *distance* and *tripDuration* (in minutes) between the previous and current stops, along with the *plannedTime* of the stop.

A *DistributionCenter* is only allowed to plan tours with the vehicle of its fleet. Further, a tour planned by a *DistributionCenter* can only deliver or transfer parcels that are currently at the *DistributionCenter*, either because they entered the study area here or they arrived by a previous transfer. The same holds for pick-ups: A tour of a *DistributionCenter* can only pick up parcels that are assigned to the distribution center. Further, the tours of a *DistributionCenters* must comply with the restrictions specified by the *RegionalReach* of the distribution center.

3.1.6 Results View Type

The results of a logiTopp simulation run are documented in a log-based format. This log comprises various events occurring during the simulation run and decisions made by models. The data model designed to capture these results is extensible, allowing for the addition of new events, decisions, and attributes with minor adjustments to the implementation. The different existing log categories are presented in tabular form.

event attribute	volatile value		description
	Private Parcel	Business Parcel	
time	-	-	*Time* of the state change.
parcelID	no	no	*Id* of the affected parcel.
producer	no	no	*Producer* (*id*) of the parcel.
consumer	no	no	*Consumer* (*id*) of the parcel.
destinationType	yes	-	Only for *PrivateParcel*: currently set *DestinationType*.
destinationZone	yes	no	*Zone* of the parcels destination (for *PrivateParcel* this depends on the *DestinationType*).
destinationLocation	yes	no	*Location* of the parcels destination (for *PrivateParcel* this depends on the *DestinationType*).
state	yes	yes	State of the parcel, possible values: *DELIVERED*, *RETURNING*, *ONDELIVERY*, *UNDEFINED*.
isDeliveryAttempt	yes	yes	True if the state changes occurred during a delivery attempt; otherwise, false.
deliveryAttempts	yes	yes	Number of previous delivery attempts.
deliveryVehicleID	yes	yes	*Id* of the current responsible *DeliverVehicle* or null.
deliveryTime	(no)	(no)	Actual time of delivery or null if not yet delivered (fixed after successful delivery).
recipientType	(no)	(no)	Describes who actually received the parcel or null (fixed after successful delivery), possible values: *PERSONAL*, *HOUSEHOLDMEMBER*, *NEIGHBOR*, *PACKSTATION*, *BUSINESS*, *DISTRIBUTION_CENTER*.

Table 3.1: LogiTopp metamodel: Results View Type - Parcel state changed log.

The parcel state changed log category captures all state changes of a parcel. An initial log entry is created when a parcel is inserted into the simulation. At this point, the state of the parcel is *UNDEFINED*. The state changes to *ONDELIVERY* when a parcel is loaded in a delivery vehicle. If the delivery was successful, the state of the parcel changes to *DELIVERED*. Otherwise, the parcel will be returned to the distribution center (*RETURNING*). In case of a transfer of a parcel between two distribution centers, log entries are also created when the parcel is loaded into the delivery vehicle and arrives at the distribution center. Then, the state of the parcel remains unchanged, except that the distribution center is the final destination of the parcel.

Table 3.1 shows the elements and semantics of the parcel state change log. The table also distinguishes between attributes that remain constant during the lifetime of a parcel and attributes that may change during the simulation (volatile attributes). These are different for business and private parcels, as the destination of private parcels can change dynamically within the simulation.

In logiTopp, private parcels can be delivered to neighbors if no member of the receiver's household is currently at home. If a delivery agent tries to deliver a parcel to a neighbor an entry in the neighbor delivery log is created. Table 3.2 shows the elements with their semantics of this log category.

A further log category captures the execution of delivery tours. Table 3.3 shows the elements with their semantics of this category. Log entries are created at every stop of a delivery tour, as well as at the start and end of a tour (*LOAD* and *UNLOAD*).

Additionally, log categories exist for private and business orders, representing parcels sent to private individuals and businesses, respectively. Another category, business production, encompasses parcels sent by businesses. Log categories also cover the selection of delivery partners and the choice of transport chains. Discussions of these categories are not included here, as the outcomes of the associated events and decisions are directly reflected in the entities and relations of the demand, transport infrastructure, and logistic solution view types.

event attribute	description
time	*Time* of the delivery attempt.
parcelID	*Id* of the parcel.
zone	*Zone* of the delivery attempt.
success	True if any of the neighbors were at home.
numberOfNeighbors	Maximum number of neighbors with whom a delivery is attempted.
checkedNeighbors	Actual number of neighbors with whom a delivery is attempted.

Table 3.2: LogiTopp metamodel: Results View Type - Neighbor delivery log.

event attribute	description
time	*Time* of the vehicle event.
CEPSP	*Id* of the responsible CEPSP.
distributionCenter	*Id* of the responsible DistributionCenter.
vehicle	*Id* of the vehilce.
event	Type of event, possible values: *LOAD*, *STOP*, *UNLOAD*.
stopNo	*LOAD* number of stops of the tour *UNLOAD* 0 *STOP* current stop number
zone	*Zone* of the current stop/*DistributionCenter*.
location	*Location* of the current stop/*DistributionCenter*.
distance	*LOAD* total distance of the tour *UNLOAD* 0 *STOP* (planned) distance between the previous stop and current stop
tripDuration	*LOAD* total duration of the tour *UNLOAD* 0 *STOP* (planned) trip duration between the previous stop and current stop
toDeliver	*LOAD* number of parcels that have to be delivered throughout the whole tour *UNLOAD* 0 *STOP* number of parcels that have to be delivered at this stop
sucessDeliver	*LOAD* 0 *UNLOAD* 0 *STOP* number of parcels that have been successfully delivered
toPickup	*LOAD* number of parcels that have to be picked up throughout the whole tour *UNLOAD* 0 *STOP* number of parcels that have to be picked up at this stop
sucessPickup	*LOAD* 0 *UNLOAD* 0 *STOP* number of parcels that have been successfully picked up
deliveryDuration	*LOAD* total delivery duration of the tour *UNLOAD* 0 *STOP* delivery duration of the stop
numReturning	Number of parcels that could not be delivered in the vehicle.
numCollected	Number of picked-up parcels in the vehicle.

Table 3.3: LogiTopp metamodel: Results View Type - Tour stop log.

3.2 MATSim-Freight

Before introducing the individual view types of the MATSim-Freight metamodel, we briefly discuss some of its cross-cutting concerns.

MATSim operates within a simulation period spanning an entire day. Absolute times are represented by the count of seconds from the day's commencement, whereas relative times denote the duration in seconds between two timestamps. The count of seconds is usually represented as a floating point number. Both absolute and relative times can denote an undefined timestamp or duration through convention, achieved by setting the value to negative infinity.

Regarding localization, MATSim supports a wide range of coordinate systems and distance units. The selected coordinate system and distance unit should align and remain consistent throughout the input data. Typical and preferred choices are cartesian coordinate systems with meters as distance units. The chosen distance unit also impacts other input data, such as velocities and lengths.

In terms of capacity and dimension specifications, MATSim uses varying representations. The metamodel contains both volume and weight specifications. MATSim-Freight uses another approach: the values used for capacities and dimensions are unit-free and only need to be in relation to other capacity and dimension values. It is up to the user or the input data to decide whether capacities and dimensions are specified as volume or weight and which unit is used.

If not explicitly modeled, monetary and economic evaluations (e.g., scores) are unit-free. However, monetary values must be considered in relation to each other.

MATSim uses the *id* attribute as a unique identifier for instances of a class, enabling distinct identification and referencing of entities within the model. In MATSim *id* attributes are represented as strings.

3.2.1 Network View Type

The MATSim network is essentially a simple directed graph comprising nodes and links that connect two nodes unidirectionally by one or more lanes. The network model is tailored to the queue-based traffic flow model, by delineating attributes like capacities, lengths, and free speeds. Figure 3.11 displays the network view of the MATSim metamodel. Subsequently, we briefly describe the elements shown:

- The enum *TransportMode* enumerates the allowed modes of transport in MATSim.
- A *Network* consists of *links* and *nodes*, with additional relevant parameters specified for the entire network. The *capacityPeriod* is given in seconds, defining

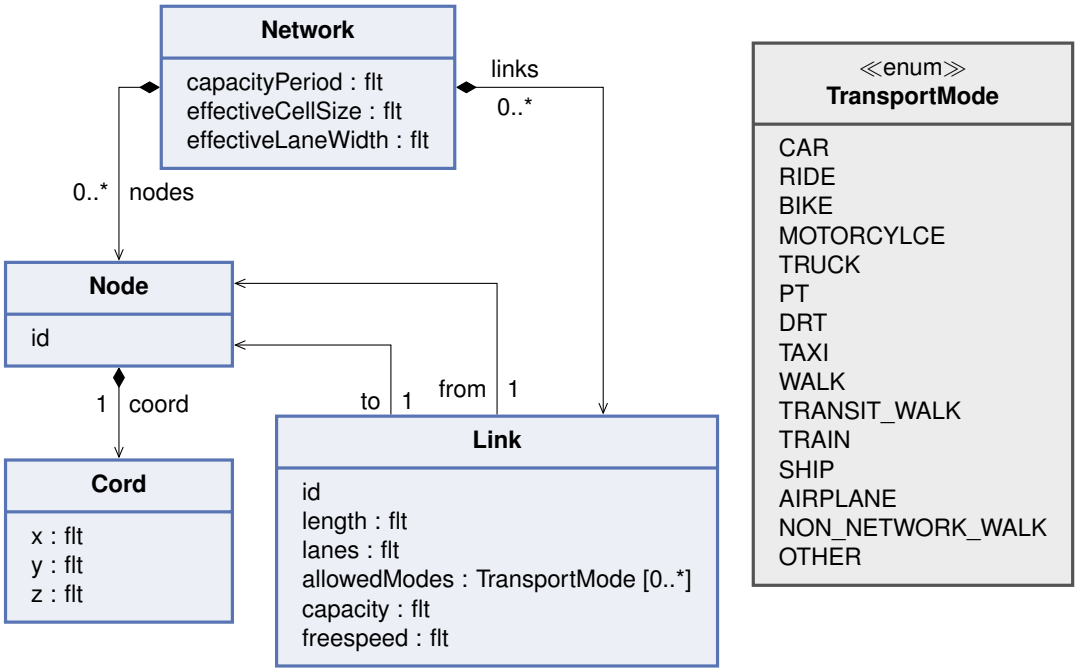


Figure 3.11: MATSim-Freight metamodel: Network View Type.

the reference value for all capacity values. *EffectiveLaneWidth* specifies the default width of all lanes in the network. The *effectiveCellSize* denotes the average space occupied by a vehicle under maximum density conditions and, in conjunction with the number of lanes, establishes the maximum density of vehicles on a network link.

- A *Node* represents a specific location or point, such as an intersection, junction, or endpoint, within the network where Links converge or originate. Every Node has an *id* and is located by coordinates (*coord*).
- Coordinates (*Cord*) are described by the *x* and *y* values, with the possibility of an additional *z* value.
- *Links* connect two nodes unidirectionally, specified by their *from* and *to* references. Each *Link* has an *id* and a *length*. A *Link* can consist of one or more lanes, with the number of lanes specified by the *lanes* attribute. Not every *TransportMode* is allowed on every Link, hence, a Link has a set of *allowedTransportModes*.

3.2.2 Population View Type

The population viewpoint focuses on the description of entities that are possibly responsible for creating any logistic demand. In the MATSim-Freight metamodel, this definition solely encompasses persons. An essential function attributed to individuals in MATSim involves their utilization of vehicles. Thus, we start by describing the metamodel for

vehicles and vehicle types. Subsequently, we proceed by describing persons, their attributes, and their affiliation with a household. It's important to note that MATSim-Freight doesn't inherently include an advanced demand generation mechanism; typically, initial demand is specified through input data in reviewed studies. Nevertheless, we opted to incorporate these entities into our analysis. The Freight Receiver contribution (discussed in Section 3.2.7) fills parts of this gap.

Figure 3.12 shows the metamodel elements related to vehicles and vehicle types. Subsequently, we briefly describe these elements:

- *Vehicles* serves as the root object and keeps track of all *vehicleTypes* and references all *vehicles* in the simulation.
- A *VehicleType* defines the properties of a vehicle, possessing an *id* and a *description*. Additionally, each *VehicleType* is linked to a *TransportMode* of the network (*networkMode*), specifying which Links can be utilized. The vehicle's dimensions are specified by its *width* and *length*, alongside the maximum velocity (*maxVelocity*). Moreover, details about the engine type and consumption (*engineInformation*), data relevant to the scoring module (*costInformation*), and information necessary for various traffic flow model implementations (*pcuEquivalents* and *flowEfficiencyFactor*) are available but not further described here.
- VehicleCapacity describes the capacity of a vehicle for transporting persons and goods. The number of persons is determined by the seating capacity (*seats*) and the amount of allowed standing passengers (*standingRoom*), e.g., in a bus. The capacity for goods transport is limited either by volume in m^3 (*volumeCubicMeters*) or by weight in tons (*weightTons*). The actual attribute used to describe capacity in MATSim-Freight is the *other* attribute, which has no fixed unit but must be in relation to other capacity and dimensional specifications.
- A *Vehicle* is identified by an *id* and described by their *type*.
- A specific instance of a *Vehicle*, relevant for the population viewpoint, is a *PersonVehicle*, describing a vehicle owned and operated by an ordinary person.

The metamodel elements related to describing persons and households are shown in Figure 3.13. Subsequently, we briefly describe these elements:

- The basic entity of the population is a *Person*, identifiable by its *id* and with no further attributes.
- A *Household* consists of a set of *Persons* (*members*). Household members can use a set of *vehicles*. The economic situation of a household is described by its *income*.
- *Income* is modeled by an amount (*income*) in a specific *currency* (currently only EUR supported) earned within a certain period (*period*).
- *Households* serve as a root object and contain all *households*.

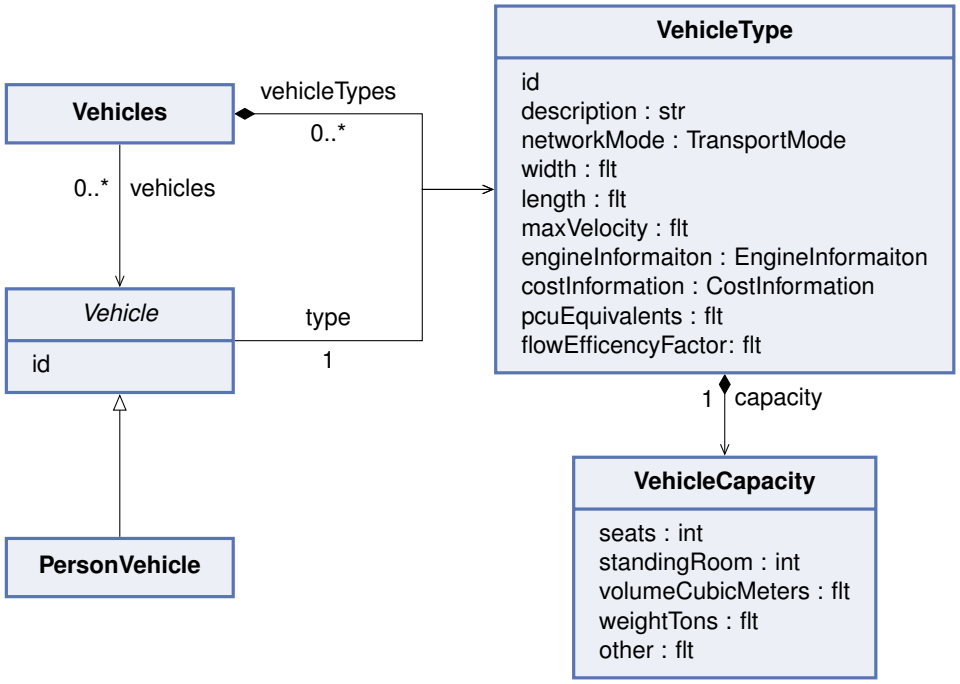


Figure 3.12: MATSim-Freight metamodel: Population View Type - Vehicles and vehicle types.

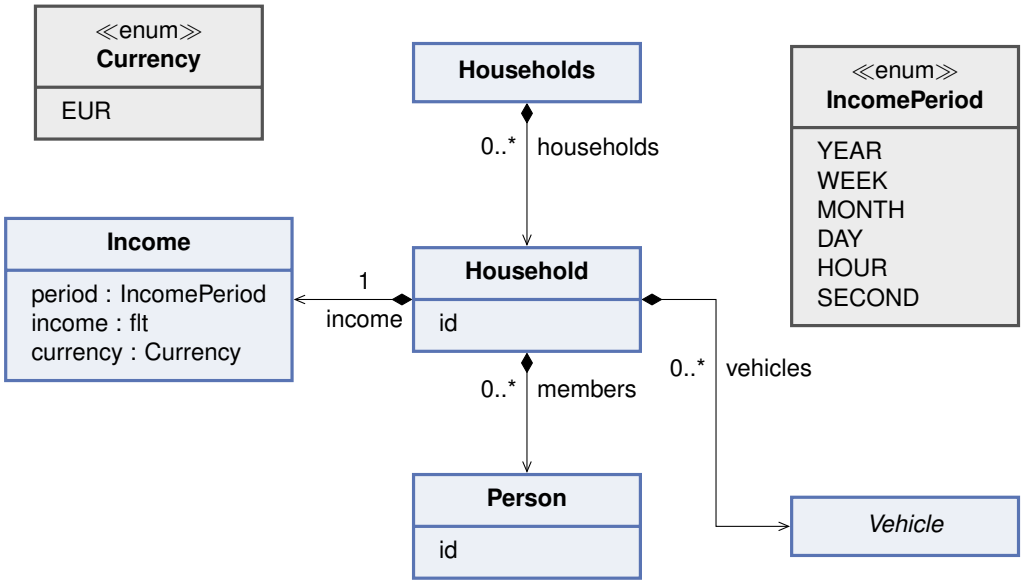


Figure 3.13: MATSim-Freight metamodel: Population View Type - Persons and households.

3.2.3 Logistic Demand View Type

In MATSim, the logistic demand is directly assigned to a carrier responsible for fulfilling this demand. This demand comprises shipments originating from and destined for locations within the study area, as well as carrier services representing shipments where either the origin or the destination lies outside the study area. Figure 3.14 shows the metamodel elements associated with logistic demand. Subsequently, we briefly describe these elements:

- A *TimeWindow* delineates a time span within the simulation period during which a logistic service must be executed, specifying its *start* and *end* times.
- *CarrierShipment* signifies the transportation of goods from an origin (*from*) to a destination (*to*) within the study area. The shipment dimension (*size*) doesn't have a predefined unit but must correlate with other dimensional and capacity specifications. Both *pickupServiceTime* and *deliveryServiceTime* designate the time taken for actual pickup and delivery at the origin and destination, respectively. Additionally, pickup and delivery times are constrained by specific time windows (*pickupTimeWindow* and *deliveryTimeWindow*).
- A *CarrierService* can be viewed similarly to a *CarrierShipment*; however, either the origin or the destination lies outside the study area, with the carrier's depot serving as the shipment's origin or destination. The dimension of the shipment, denoted by the *capacityDemand* attribute, has no predefined unit but is relative to other capacity and dimension values. The *location* indicates where the shipment needs to be picked up or delivered. The *serviceDuration* specifies the time taken by the carrier for pickup or delivery, and the *timeWindow* restricts when the carrier must carry out the pickup or delivery.
- Within the logistic demand view type, a *Carrier* is a unit responsible for handling a series of orders. These orders are divided into *CarrierServices* (*services*) and *CarrierShipments* (*shipments*).

Furthermore, there is a root element (*Carriers*) that contains all Carrier instances.

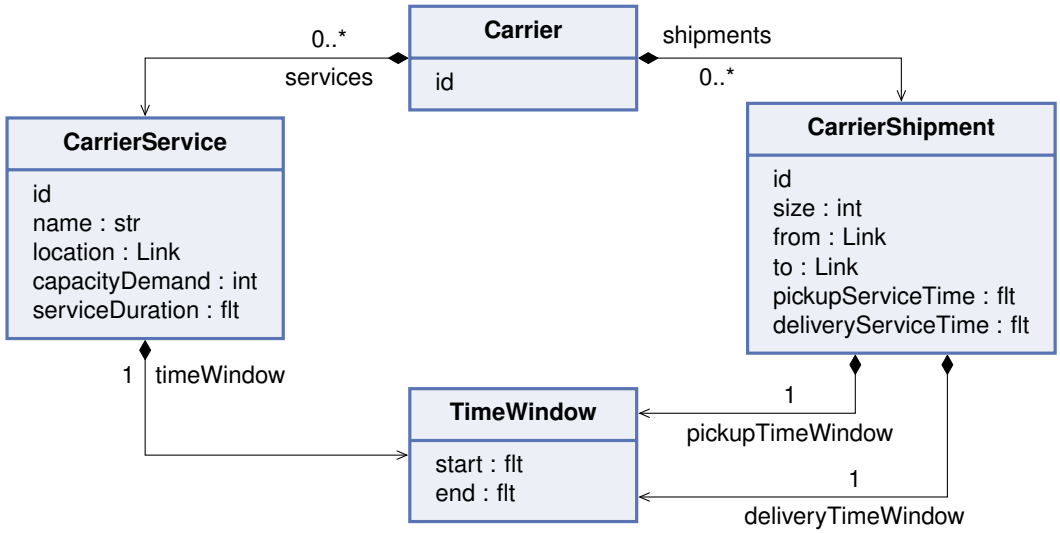


Figure 3.14: MATSim-Freight metamodel: Logistic Demand View Type.

3.2.4 Transport Infrastructure View Type

The available logistic transport network in MATSim primarily comprises carriers that operate a fleet of vehicles. These vehicles can utilize the public transport network. Implicitly modeled depots only function as origins and destinations for tours executed by carrier vehicles. A concept of goods handling and, thus, the formation of transport chains does not exist and can only be modeled implicitly. Figure 3.15 shows the metamodel elements associated with transport infrastructure. Subsequently, we briefly describe these elements:

- A *Carrier* is a business providing logistic services and thus has capabilities to provide these services (*carrierCapabilities*).
- *CarrierCapabilities* primarily comprise the fleet of vehicles operated by the carrier. This includes the various *VehicleTypes* within the fleet, specified in the *vehicleTypes* set. The fleet can be modeled as either *FINITE* or *INFINITE* (*fleetSize*). For an *INFINITE* fleet, the simulation can employ an unlimited number of vehicles from the defined *vehicleTypes*. In the case of a *FINITE* fleet, specific available vehicles are explicitly modeled (*carrierVehicles*).
- A *CarrierVehicle* is a concrete *Vehicle* within the carrier's fleet. In addition to the properties of an abstract *Vehicle*, it has a *location* indicating the depot's location or the vehicle's base, where tours must commence and conclude. Further, the operational timings are constrained by the *earliestStartTime* and *latestEndTime*, allowing to express further limitations such as working-hour restrictions.

Furthermore, there is a root element (Carriers) that contains all Carrier instances.

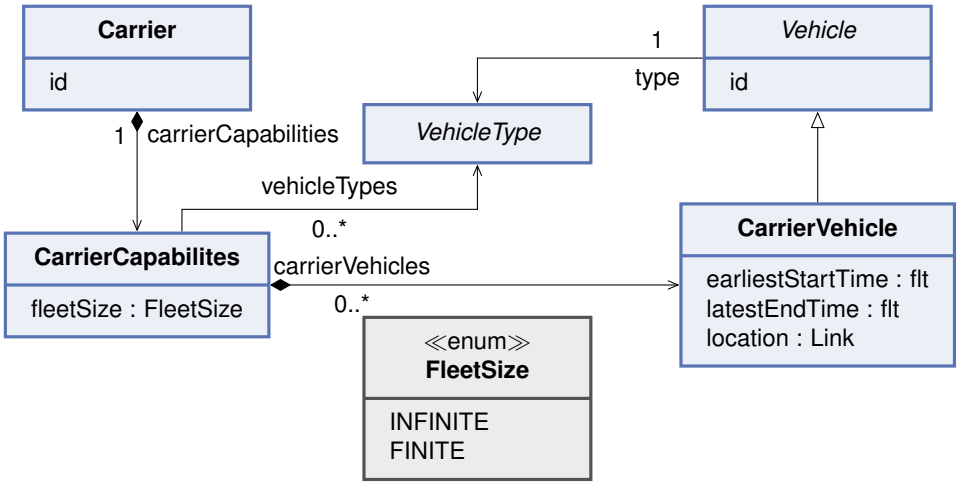


Figure 3.15: MATSim-Freight metamodel: Transport Infrastructure View Type.

3.2.5 Logistic Solution View Type

The logistic solution view type contains elements that describe how the logistic demand is handled. Following the paradigm of MATSim, each carrier devises and optimizes a plan to handle its demand during the simulation. This plan comprises a set of tours executed by the carrier's vehicles within specific timeframes. Each tour is a sequence of activities interconnected by legs. These activities primarily involve the pickup or delivery of shipments. We commence by describing the metamodel elements related to describing a tour and subsequently describe the carrier's plan.

Figure 3.16 shows the metamodel elements used for tour modeling. A Tour is identified by an *id* and consists of a list of *TourElements* (*tourElements*). These *TourElements* can either be a *TourActivity* or a *Leg*. A *TourActivity* represents a stop within the tour, including the tour's start and end. Consequently, *TourActivities* and *Legs* must alternate within the *tourElements* list, with *Legs* linking the preceding activity to the subsequent one. The initial item in the *tourElements* list must be a *Start* activity, and the final item should be an *End* activity. Both the *Start* and *End* activities are explicitly referenced by a *Tour* via the *start* and *end* attributes. We now shortly describe the different *TourActivities* and the *Leg* modeling:

- A *Leg* connects two *TourActivities*. It contains the chosen *route*, whose modeling we will not go into any further. It also includes the expected travel time for the route (*expectedTransportTime*) and the expected time for the start of the leg (*expectedDepartureTime*).
- A *ServiceActivity* is a *TourActivity* that describes the carrying out of a *CarrierService* (*service*). Additionally, it specifies the *expectedArrivalTime* at the stop.
- *Delivery* and *Pickup* are both *ShipmentBasedActivities*, which are also subtypes

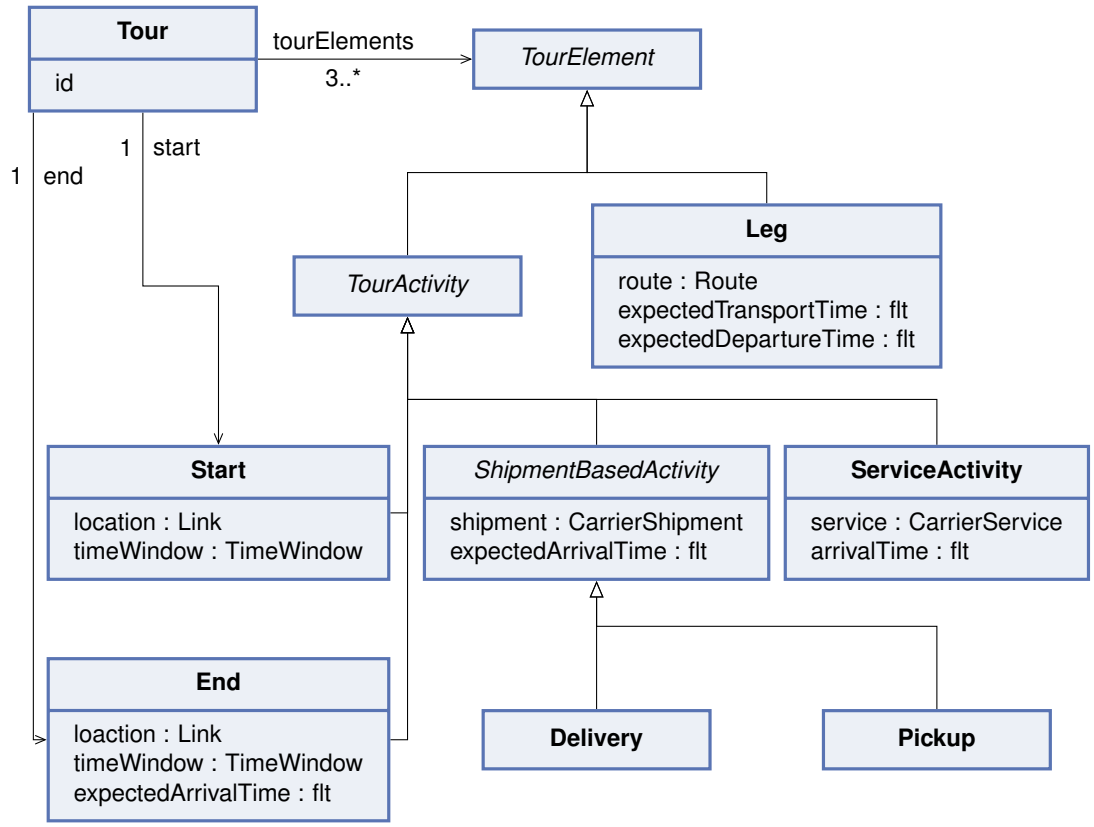


Figure 3.16: MATSim-Freight metamodel: Logistic Solution View Type - Tours.

of *TourActivities*. They relate to the handling of a *CarrierShipment* (*shipment*). As in a *ServiceActivity*, the *expectedArrivalTime* is specified.
- Special kinds of *TourActivities* are the *Start* and *End* activities. Both have a *location* representing the tour's start and end. These locations must be consistent with the location of the *CarrierVehicle* assigned to carry out the tour. The *timeWindow* describes the allowed time window for the start or end of the tour, usually limited by the earliest start and latest end time of the assigned *CarrierVehicle*. Additionally, the *End* activity contains information about the expected time for the end of the Tour (*expectedArrivalTime*).

The specified times within a *Tour* have to be consistent. The time required for a *TourActivity* is defined by the related *CarrierShipment* (*pickupServiceTime*, *deliveryServiceTime*) or *CarrierService* (*serviceDuration*). The travel time between stops is indicated by the *expectedTransportTime* of the respective *Leg*. An initial start time within the *timeWindow* of the *Start* activity can be selected and set as the *expectedDepartureTime* of the first *Leg*. The *expectedArrivalTime* of each activity can be derived from the *expectedDepartureTime* and the *expectedTransportTime* of its preceding *Leg*. Similarly, the *expectedDepartureTime* of a *Leg* can be calculated from the *expectedArrivalTime* and the time required at its previous *TourActivity*. Furthermore, it is required for the *expectedArrivalTime* of the *End* activity to fall within its specified *timeWindow*.

The metamodel elements utilized to describe a carrier's plan, vehicle assignments to tours, and tour schedule are illustrated in Figure 3.17. Subsequently, we briefly describe these elements:

- A *CarrierPlan* delineates a set of activities, their sequence, schedule, and the utilization of capabilities to fulfill the demand of a Carrier. This is manifested as a collection of *scheduledTours*. Each tour within these must contain corresponding activities for every demanded *CarrierService* and *CarrierShipment*. For the use of MATSim's co-evolutionary algorithms, a *score* of the plan is provided.
- The allocation of a vehicle from the respective Carrier and a departure time to a tour is modeled by a *ScheduledTour*, containing attributes such as *tour*, *vehicle*, and *departureTime*.
- A Carrier is the entity responsible for fulfilling logistic demands and can rely on specific capabilities. To do so, a *Carrier* possesses a *CarrierPlan*. The co-evolutionary algorithm of MATSim aims to optimize these plans for all agents, requiring each agent to maintain a set of *plans* and a reference to the currently *selectedPlan*.

Furthermore, there is a root element (*Carriers*) that contains all *Carrier* instances.

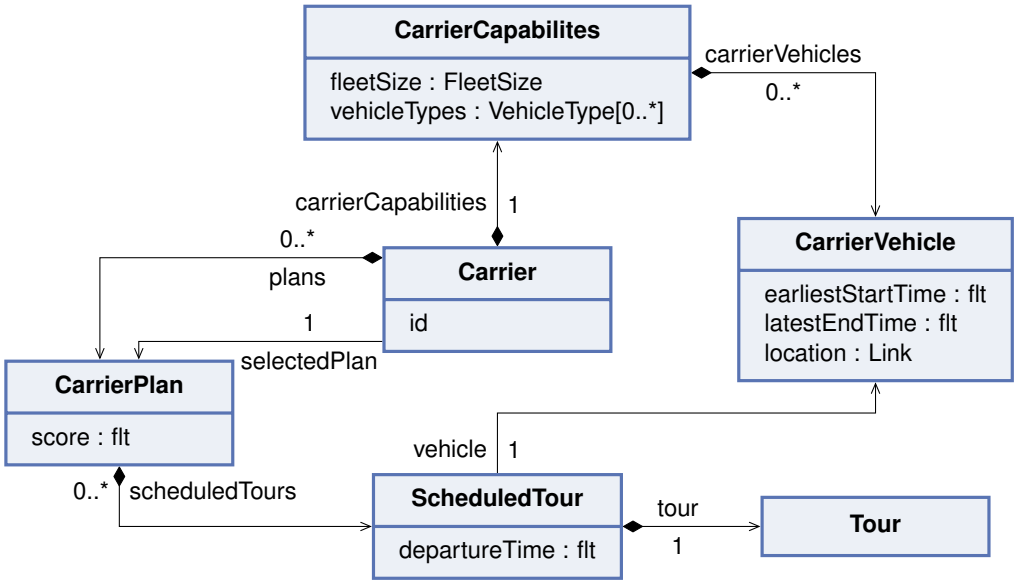


Figure 3.17: MATSim-Freight metamodel: Logistic Solution View Type - Carrier plan.

3.2.6 Results View Type

The results of a simulation in MATSim-Freight are recorded in an event-based format. Figure 3.18 shows the events introduced by the MATSim-Freight contribution. These focus solely on the execution of delivery tours. It is possible to employ other events from the MATSim framework to analyze the results of a simulation run. Here, we provide a brief description of the events within the MATSim-Freight contribution:

- An *Event* is a generic, abstract MATSim event that occurred at a specific *time* during a simulation run.
- The abstract *CarrierEvent* is an *Event* and serves as the supertype for all kinds of events describing the execution of delivery tours. It includes the associated *carrier* and *carrierVehicle*. Additionally, it states the location of the vehicle (*link*) at the time the event occured
- *CarrierTourEvents* describe the start (*CarrierTourStartEvent*) and end (*CarrierTourEndEvent*) of a tour's execution. They contain a reference to the corresponding *tour*.
- A *CarrierServiceEvent* is used to describe the execution of a *CarrierService*. Both the beginning and the end of a stop, related to a *CarrierService*, are modeled by the *CarrierServiceStartEvent* and *CarrierServiceEndEvent*, respectively. A *CarrierServiceEvent* includes a reference to the executed *carrierService*, its duration (*serviceDuration*), and shipment dimension (*capacityDemand*).
- The execution of a *CarrierShipment* is depicted by a *CarrierShipmentEvent*. Separate events exist to describe the start and end of both pickup and delivery (*CarrierShipmentPickupStartEvent*, *CarrierShipmentPickupEndEvent*, *CarrierShipment-*

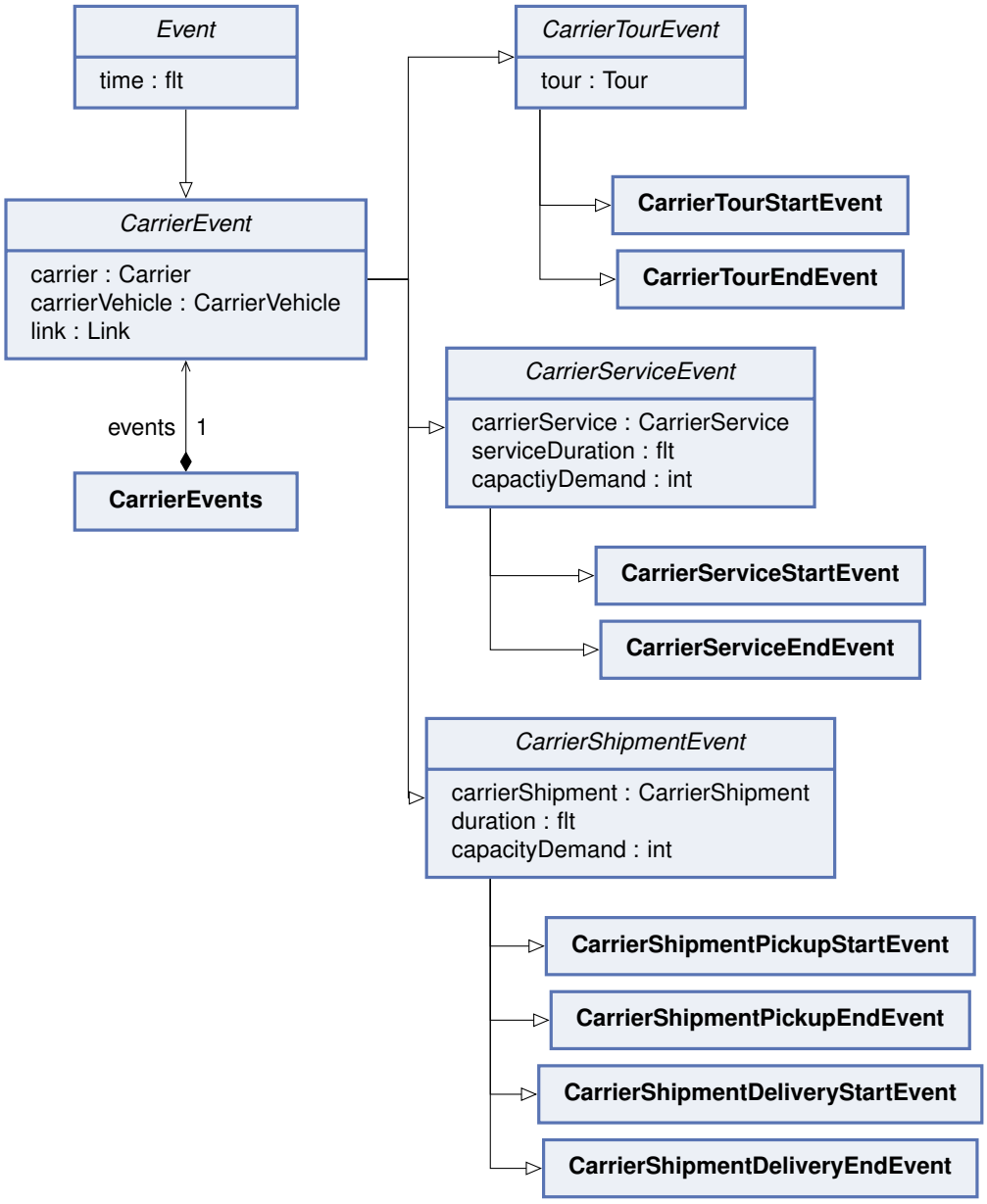


Figure 3.18: MATSim-Freight metamodel: Results View Type.

DeliveryStartEvent, *CarrierShipmentDeliveryEndEvent*). A *CarrierShipmentEvent* contains a reference to the corresponding *carrierShipment*, the *duration* of the pickup or delivery, as well as the shipment dimension (*capacityDemand*).
- *CarrierEvents* is a root element, that contains all *events* occurred durring a simulation run.

3.2.7 Freight Receiver View Type

The freight receiver contribution [10] extends the existing metamodel by introducing additional elements. The primary entity is the *Receiver*, which expresses its demand for specific *ProductTypes* through *ReceiverProduct* instances. The actual demand of a receiver can be modeled interchangeably, and our metamodel incorporates the currently available *SSReorderPolicy*. There, the weekly demand for a product type is determined as the difference between the maximum inventory level (*maxLevel*) and the current inventory level (*stockOnHand*). Demand only exists if the current inventory level (*stockOnHand*) is below the minimum (*minLevel*). This weekly demand is then evenly divided across several deliveries a week (*numberOfWeeklyDeliveries*). However, MATSim supports only a simulation period of one day. Thus, the *numberOfWeeklyDeliveries* has been fixed to 5 days in the current implementation. The *Receiver* has a *ReceiverPlan* consisting of a set of *ReceiverOrders* to fulfill this demand. The *ReceiverOrders* assigns a carrier and a delivery time window to a set of orders. Figure 3.19 shows the metamodel elements introduced by the freight receiver contribution. Subsequently, we briefly describe these elements:

- The *Receivers* class acts as the root object and contains a set of *receivers* and available *productTypes*.
- A *ProductType* represents a specific type of good that can be transported. It has an *id* and a *description*. Every *ProductType* has a *location* from where it has to be delivered to the *Receiver* (*originLink*). Further, the required transport capacity per unit of the product is given (*transportCapacity*). However, the unit of the transport capacity is not predefined but must be in relation to other capacity and dimensional values as it gets transferred to the size (*other*) attribute of a *CarrierShipment*.
- *Receivers* are entities that order products. A *Receiver* has an *id* and a single *location*. Its demand is determined by a set of *ReceiverProducts* (*products*). To fulfill its demand, the *Receiver* places orders and commissions carriers. The entire quantity of orders and delivery assignments is referred to as the plan of a *Receiver*. The co-evolutionary algorithm of MATSim aims to optimize these plans for all agents, requiring each agent to maintain a set of *plans* and a reference to the currently *selectedPlan*.

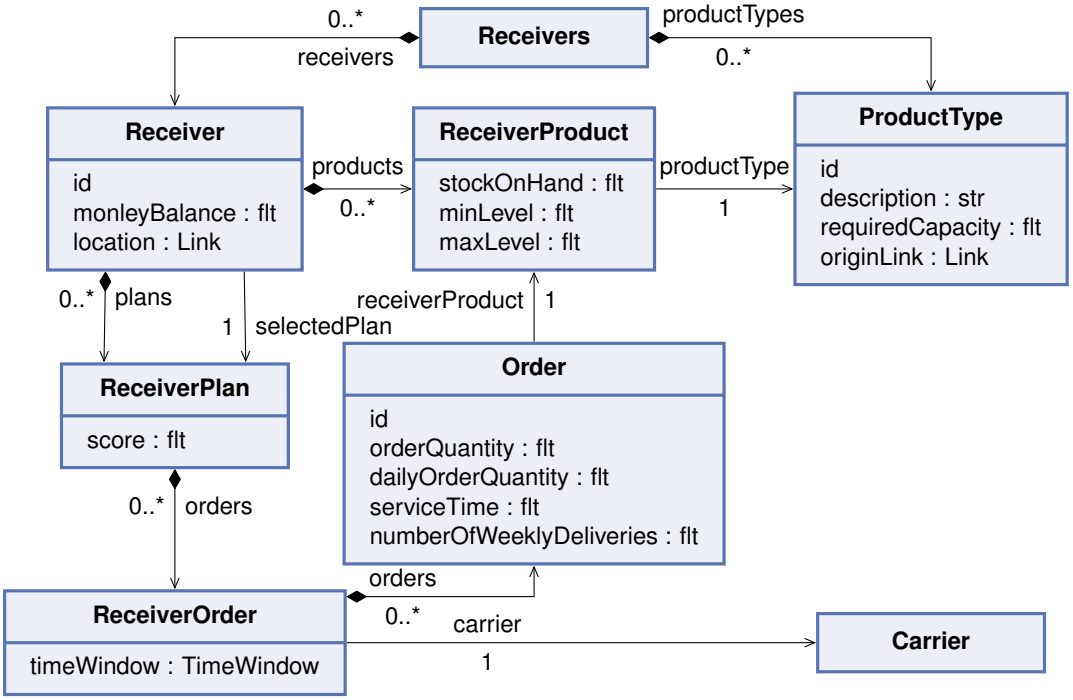


Figure 3.19: MATSim-Freight metamodel: Freight receiver contribution.

- *ReceiverProducts* represent the demand of a single *Receiver* for a specific *ProductType* (*productType*). The *Receiver* has an initial quantity of the ProductType in stock (*stockOnHand*). Furthermore, his storage capacity is limited (*maxLevel*), and an amount of articles is defined, which should not be undercut (*minLevel*). As soon as this is reached, a reorder must be placed.
- A *ReveiverPlan* is a set of *ReceiverOrders* (*orders*). For the use of MATSim's co-evolutionary algorithms, a *score* of the plan is provided.
- A *ReceiverOrder* assigns a set of *orders* to a *carrier* and defines a *timeWindow* within which the *Carrier* should deliver the orders.
- An *Order* is placed by a *Receiver* and refers to a single *ReceiverProduct* of the Receiver (*receiverProduct*). It has an *id* and a weekly quantity of ordered units (*orderQuantity*). The *dailyOrderQuanity* is derived from the *orderQuantity* and the *numberOfWeeklyDeliveries*. Further, the time required for the delivery of the order at the receiver is given (*serviceTime*).

This modeling allows a *Receiver* to develop very flexible plans, e.g., by distributing the deliveries of a single *ProductType* to different *Carriers* and time windows.

3.3 Other Metamodels

In this section, we present selected concepts of other freight transport models, namely FREMIS [50] and SimMobility Freight [51], that differ from or complement the concepts presented so far. We refrain from presenting the complete metamodel with all its elements and attributes and instead present the concepts from a high-level perspective. Section 3.3.1 starts with concepts from FREMIS, and Section 3.3.2 presents concepts from SimMobility Freight.

3.3.1 FREMIS

Population View Type:

FREMIS has a two-level structure to represent business entities. Business establishments represent organizations at a specific location that produce, process, and store commodities or provide business or logistic services. To achieve this, a business can have various facilities. In the population view, these are commodity production and business service facilities. A single commodity production facility has a production capacity and productivity. It is capable of producing a single product or can process multiple commodities interchangeably. The same applies to business service facilities. These have a service provision capacity and can provide one or multiple types of services. The second level of the business structure is a firm that owns and operates one or more business establishments. Regarding model semantics, this means that business establishments within a firm are more likely to interact with each other.

Further entities of the population in FREMIS are end consumers. Besides private persons that are organized in households, FREMIS explicitly states the government in this category. In the demand generation process, end consumers initiate the demand for business services and commodities. This demand leads to the production of products and services, which in turn creates demand for the products and services required by a business establishment to produce these products. All these processes potentially require shipment and lead to the generation of logistic demand.

Logistic Demand View Type:

To create logistic demand, firms and business establishments make several decisions that result in a set of contracts. A business establishment can have multiple commodity and business service contracts with other business establishments about the provision of business services and commodities. A commodity contract has a vendor, a customer, a price, and a list of shipments that are shipped between the vendor and the customer. A business service contract is modeled similarly, but instead of a list of shipments, it defines the business services that the vendor provides to the customer. Shipments

are commodity movements from an origin business establishment to a destination business establishment or an end consumer. A shipment is further specified by its date, weight, type of commodity, and price. Optionally, time windows for delivery and pickup, as well as special handling requirements, can be specified. Business establishments can act at the same time as customers and vendors, allowing the representation of complex business relations and commodity flows. While it is clear how commodity contracts result in demand for transportation, only some kinds of business services require transportation, e.g., on-site visits to the customer's establishment. Therefore, FREMIS only focuses on business services that require transportation and further assumes that the business establishment's fleet is capable of handling the required amount of transportation.

Transport Infrastructure View Type:

FREMIS does not distinguish between regular firms and transportation service providers. Thus, a business establishment can also have a logistic facility. A logistic facility can operate vehicles of different types, transshipment centers, warehouses, and intermodal terminals. All these resources are located at the location of the associated business establishment. Further, a business establishment may have logistic service contracts that commission other companies or business establishments to handle shipments. A logistic service contract consists of a business establishment responsible for shipments, a logistics business establishment that executes shipments, the list of shipments to be handled, and a price.

This model allows the representation of a wide range of company constructs. From regular firms that buy logistic services over own-account carriers that operate a private fleet to specialized transport service providers. It is also possible to represent companies that act only or partially as brokers and offer logistics services but delegate these to third parties through subcontracts. The companies, with their business establishments and associated logistical facilities as well as their contractual relationships, then represent the transport network.

Another concept is the possibility for companies to own and use parts of the network exclusively. Companies can exclusively own network links. This, for instance, can represent own railway tracks or even pipelines.

3.3.2 SimMobility Freight

Logistic Demand View Type:

Comparable to FREMIS, the logistic demand in SimMobility Freight is determined and represented in a three-level process. In the first level, the estimated annual quantity

(e.g., weight, number of units) an establishment receives or produces of a commodity type is represented. The second level breaks down this demand into one or more contracts. Contracts represent supplier-receiver pairs and specify the amount quantity covered by the product. In the third level, the contracts are translated into one or more shipments of a specific size.

Transport Infrastructure View Type:

A distinctive feature of SimMobility Freight is the representation of overnight parking choices. While most of the investigated models assume that tours start and end at a depot, tours in SimMobility could also start and end at other parking facilities. A vehicle can be parked in the depot, in public spaces, in parking spaces provided by other companies, or in publicly accessible parking spaces.

To determine a parking choice, the network and population views must be extended by information about available parking spaces, their capabilities, and costs. SimMobility Freight is capable of evaluating the effects of policies related to parking. Thus, these must also be modeled.

Logistic Solution View Type:

The logistic solution view of SimMobility Freight, like the already presented model, uses a tour-based modeling approach. A novelty is the consideration of parking choices within the tour. The parking choice is based on the availability of suitable parking infrastructure and effects such as waiting time. Possible parking choices are (un)loading bay area, public car park, and illegal on-street parking.

Another novel concept of the SimMobility Freight metamodel is shipment records. Besides just modeling the delivery tours, the planned sequence of a shipment's transport is explicitly represented in a shipment record. Shipment records and related tours are linked and consistent. In the metamodels presented so far, the information of a shipment record could potentially be extracted from the planned tours or in a log-based result view derived from the event sequence. However, these result-based shipment records usually represent the actual shipment sequence of the simulation, while this shipment record represents the initially planned shipment sequence.

3.4 Comparison

In Sections 3.1, 3.2, and 3.3, we introduced the metamodels of logiTopp and MATSim, along with selected concepts from other metamodels. In this section, we then compare the logiTopp and MATSim-Freight metamodels, occasionally highlighting distinctions from other presented metamodels. The focus of this section is on major differences and

similarities. We start by briefly comparing the model steps and cross-cutting concerns, and thereafter, we compare each view type of the two metamodels.

The first major difference lies in the model steps and simulation process employed by logiTopp and MATSim. Both were described in detail in Section 2.3.2. While logiTopp is structured in a long-term and short-term module with dynamic planning and execution of logistic processes, MATSim-Freight tries to create and iteratively optimize a logistic solution for a given logistic demand using a co-evolutionary algorithm. Thus, the MATSim metamodel contains concepts supporting this co-evolutionary algorithm, which involves a set of scored plans and a selected plan for each entity. The scores can be seen as economic values, which are only relative and unit-free. Other metamodels may also represent economic or monetary values, e.g., to simulate and describe contracts and use other modeling, but this is not further discussed in this work. Note that logiTopp's metamodel does not represent the concepts supporting its dynamic planning and simulation approach, also because we omitted representing state-describing properties in the metamodels.

Regarding cross-cutting concerns, a major difference between logiTopp lies in the simulated time period. Which is typically a day in MATSim and a whole week in logiTopp. Even if the exact representation differs, time information can be easily converted, with the exception of the weekday component, as the granularity is in seconds. Both logiTopp and MATSim are not constrained to a specific coordinate system but prefer Euclidian coordinate systems, which have meters as the typical base unit. In terms of dimension and capacity values, various approaches exist. LogiTopp uses volumes, while other metamodels employ weights or both. MATSim-Freight does not have defined units for dimensions and capacities.

3.4.1 Comparing Network View Types

Both logiTopp and MATSim-Freight utilize directed graphs to represent their network structures. LogiTopp introduces an additional reference to the opposite edge, although this information can be derived from the existing network.

However, variations exist in both models' attributes associated with nodes, edges, and the network. MATSim, with its queue-based traffic simulation, features specific attributes like *capacityPeriod*, *effectiveCellSize*, and *effectiveLaneWidth* for network representation. It also considers a link's number of lanes, *capacity*, and *freespeed*, distinguishing between various transport modes with assigned sets of allowed modes. All these concepts are not included directly in logiTopp's metamodel but may be considered by the external tool that calculates the travel times assumed in the planning and simulation step of logiTopp.

An exclusive concept in logiTopp is the use of zones, which are prominently featured in mobiTopp. In logiTopp, zones only serve the purpose of describing the extent of the service area covered by a distribution center.

A notable difference lies in the localization of entities within the network. While in MATSim, locations are defined by referencing a network link without specifying the precise location along the link, logiTopp provides detailed specifications. LogiTopp specifies the position of an entity along the edge (or link) and provides coordinates, indicating the exact position and point in the network from which the position can be accessed. This information is consolidated in an additional class, namely *Location*.

3.4.2 Comparing Population View Types

The fundamental structure for describing private persons in logiTopp and MATSim is quite similar: Both organize private persons into households. Notable differences include the representation of household locations. In logiTopp, each household is associated with a zone and a specific location, whereas this information is not explicitly modeled in MATSim. However, inferring this data from a person's activity schedule might be possible.

Regarding attributes, both metamodels capture the economic status of private persons through their income. In logiTopp, each person has an individual income, while in MATSim, income is aggregated at the household level. LogiTopp introduces a variety of attributes describing individual characteristics, such as shopping behavior. Note that some of these attributes are derived from additional data structures, like activity schedules, but are included directly as attributes in the presented metamodel for simplicity. The attributes presented here are, therefore, only a subset of the potential information available for demand generation.

Another difference lies in the presence of private vehicles in the metamodels. While vehicle ownership and properties are incorporated in logiTopp (and mobiTopp), we have excluded them from the metamodel for two reasons. First, private vehicles in logiTopp currently have no impact on freight transport and demand simulation. Second, unlike MATSim, private vehicles do not share properties with the vehicles used by the transport infrastructure.

LogiTopp's population view type encompasses businesses, incorporating a few categorizing properties and modeling opening hours. In MATSim-Freight, a business is not explicitly defined and may only exist implicitly as a location where shipments are delivered to or from. However, the freight receiver contribution introduces receivers, potentially representing businesses. Although both involve a location, the properties of receivers and businesses differ. MATSim's freight receiver contribution explicitly models

a receiver's demand, while in LogiTopp, demand is derived from its properties. Other freight transport models, such as FREMIS, undertake a significantly more complex modeling of businesses. FREMIS features a two-level structure of firms and business facilities, providing detailed information on the capabilities of each business facility. This allows for the modeling and simulation of complex supply chains and contractual relations.

These disparities in the level of detail and the included entities arise from the diverse features of the examined freight transport models in the demand generation step. While MATSim generally takes demand as a given input, other models employ advanced demand generation or even detailed market simulation.

3.4.3 Comparing Logistic Demand View Types

In the logistic demand view types, all demand for the transportation of goods and parcels is included. MATSim distinguishes between services and shipments, indicating whether the shipment is entirely within the study area or if only one destination or origin is in the destination area. Conversely, logiTopp distinguishes between business and private parcels, denoting the type of receiver or sender of the parcel. The main differences lie in modeling individual concepts or properties of the shipments or parcels, which are mostly independent of the previously mentioned distinctions. Key distinctions include:

- Origin/Destination specification: LogiTopp explicitly includes the producer and consumer of a parcel, allowing derivation of the parcel's origin and destination. In MATSim, only the origin and destination location are specified, with no reference to the entity consuming or producing the parcel.
- Supported Origin/Destination relations: LogiTopp supports a subset of consumer-producer relations, while MATSim allows shipment between any type of location.
- Delivery/Pick-up time windows: LogiTopp lacks explicit delivery and pick-up times, with these times modeled implicitly and determined dynamically during the simulation, e.g., by opening hours of businesses or the activity schedule of private persons. MATSim, on the other hand, features explicit time windows.
- Delivery/Pickup-Service times: LogiTopp determines service times dynamically by an exchangeable and dynamic model, while MATSim explicitly models service times.
- Size and weight: Both metamodels use different concepts to describe the size and weight of shipments, as already discussed in the cross-cutting concerns.

The freight receiver contribution of MATSim includes additional metaclasses within the definition of the demand viewpoint. It explicitly models a receiver's demand, translating

it into orders and subsequently transferring these orders to shipments.

Similar to MATSim's freight receivers contribution, other freight transport models also adopt a multi-level representation of demand. However, the lowest level always consists of shipments, and their modeling has no significant difference.

3.4.4 Comparing Transport Infrastructure View Types

The comparison of the transport infrastructure view types shows that the basic structure is similar. Both models incorporate multiple carriers with fleets of vehicles. However, distinctions arise in the assignment of vehicles: MATSim directly assigns fleets to carriers, with each vehicle having its own location. Conversely, logiTopp associates vehicles with distribution centers and their locations. SimMobility uniquely models overnight parking choices, confined to distribution centers in logiTopp and represented by vehicle locations in MATSim.

LogiTopp introduces the concept of limited service areas to distribution centers, lacking a direct counterpart in MATSim. Distribution centers can participate in transport chains, constituting a transport network — a feature absent in MATSim.

To overcome this difference, multiple shipments forming a chain in space and time could be created to represent transport chains despite these limitations in MATSim. Thus, the choice of transport chains and their time dimension would be predefined in the demand representation, limiting the degrees of freedom of the planning and optimization algorithms. A further concept that is only included in logiTopp is the timetable.

Another distinction lies in carrier choice. MATSim directly assigns shipments to carriers, while logiTopp dynamically determines the responsible CEPSP during simulation, influenced by contractual relations defined between businesses and CEPSPs. FREMIS does not differentiate between firms and carriers, allowing each firm to possess or lack logistic capabilities. Shipments are then distributed among various firms through a potentially complex network of shipping contracts.

Regarding vehicle modeling, logiTopp includes the vehicle type, its capacity, and the latest return time. MATSim shares similar properties and additionally includes the earliest start time. Moreover, MATSim provides a more detailed specification of vehicle types.

3.4.5 Comparing Logistic Solution View Types

The representation of logistic solutions is very similar in logiTopp and MATSim. Both metamodels utilize a tour-based structure, where each tour comprises a list of stops or tour elements, is scheduled, and is associated with a specific vehicle.

However, differences arise in the specific modeling of stops or tour elements. MATSim distinguishes between various tour activities and explicitly models legs in between, incorporating an explicit representation of the taken routing in the network—elements lacking an equivalent in logiTopp. Although both metamodels provide details about planned times, travel times, stop times, and location, this information is distributed differently across various metaclasses and their referenced elements.

A notable distinction lies in logiTopp's capability to aggregate multiple pickups and deliveries at a single stop. In contrast, in MATSim, each pickup or delivery requires its own tour activity, excluding the start and endpoint of a tour.

Although both metamodels are very similar, the solution space differs significantly. LogiTopp allows parcel transfer at distribution centers, enabling the realization of transportation of a single parcel by utilizing multiple tours and transport modes. In contrast, MATSim processes the entire shipment within a single tour. Other metamodels, such as SimMobility freight, explicitly model parcel transfers by maintaining a shipping record for each parcel.

3.4.6 Comparing Results View Types

LogiTopp and MATSim both contain a log-based format that captures the simulation trace. However, the results view types differ partially in which processes are logged and in terms of the level of detail and contained properties of the logs.

The shared subset of both is the representation of the progress in the execution of the tours. This structure is quite similar, as each metamodel contains at least one log entry for every executed stop of the tour. In contrast to logiTopp, MATSim differentiates between the beginning and end of a stop. The resulting logs further differ in structure and attributes because of the differences in allowed processes and the modeling of logistic demand and logistic solutions. Further, there are differences in which attributes are captured in the logs.

Note that only minor modifications are needed to add further log events or add attributes to existing log entries and thus align both metamodels more closely.

4 Conception

In this chapter, we present a concept that allows coupling and data exchange between several agent-based freight transport models. The primary goal of this concept is to make use of the variety of functions of several agent-based freight transport models in an integrated and combined manner. The concept is based on the idea of a common metamodel for the freight transport domain and the use of model transformations to transfer between freight transport models and common metamodel. This chapter provides an overview of the general concept. The developed common metamodel, along with the required model transformations, are then presented in Chapter 5.

This chapter is structured as follows: We begin by presenting the architecture of the concept and discuss design considerations as well as the involved components in Section 4.1. Section 4.2 then defines the assumed model steps and discusses points in the assumed model process where coupling can take place (referred to as exchange points). Following this, in Section 4.3, we discuss how the common metamodel and the entire architecture can handle variability in the employed concepts of the freight transport models.

4.1 Architecture

In this section, we present the conceptual architecture developed for the coupling of multiple agent-based freight transport models. We commence by outlining general considerations and providing an overview of the architecture, followed by a detailed examination of each component.

Figure 4.1 offers a comprehensive overview of the developed architecture. Metamodels are represented by blue boxes, while blue arrows signify model-to-model transformations between them. The green boxes denote the coupled freight transport models. The coupling process between two models typically involves several steps: initially, the requisite data for coupling is extracted from the originating model into an instance of a metamodel reflecting its data structure (simulation-based metamodel). Subsequently, model transformations from the origin simulation-based metamodel to the common metamodel and from the common metamodel to the destination simulation-based meta-

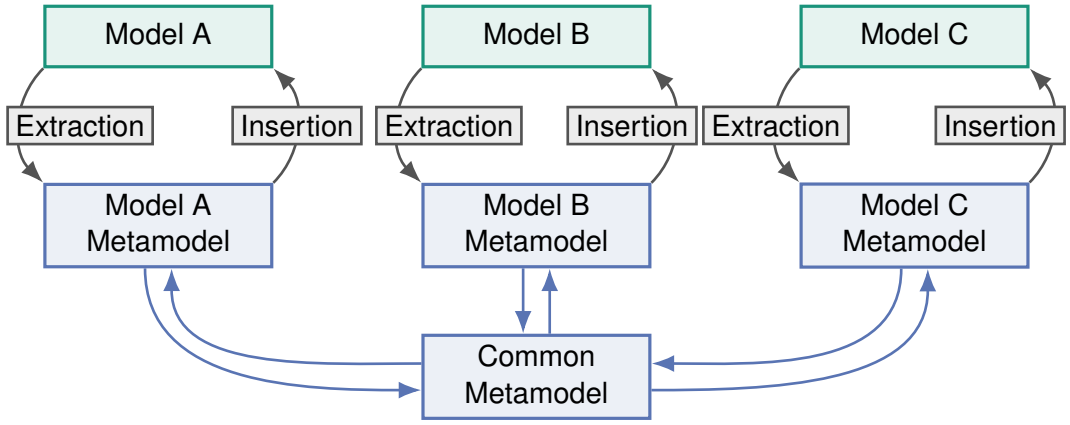


Figure 4.1: Overview of the proposed architecture for the coupling of freight transport models through a single common-metamodel.

model are applied. Finally, the resulting instance of the destination simulation-based metamodel is inserted into the destination freight transport model.

The main idea of this work was to develop and employ a common domain model, referred to as the common metamodel, comprising generally valid or widely accepted and shared domain concepts to facilitate the coupling process. The utilization of such a model yields manifold benefits. It fosters the establishment of a common language and perspective of the domain among all stakeholders, enhances knowledge sharing, communication, and collaboration, and serves as a standard for data exchange and problem definitions.

The common metamodel serves as a central component within the architecture. Model transformations are only necessitated between the common metamodel and each integrated freight transport model. Consequently, only two additional model transformations need to be established when integrating a new transport model into an existing system, effectively reducing the number of required transformations from $O(n^2)$ to $O(n)$. Moreover, for changes to already integrated models, only the transformations between the modified model and the common metamodel need revision.

Another advantage of a central model is the ability to define operations required by numerous transformations or applications on the common metamodel, thus enabling their shared utilization across multiple metamodels. This principle is employed, for instance, in handling variability, as elaborated in detail in Section 4.3.

We have opted for an explicit metamodel representation of a freight transport models data model (simulation-based metamodel) rather than a direct transfer between a freight transport model and the common metamodel. This decision is twofold: first, employing explicit metamodel representations enables the utilization of methods from model-driven engineering, resulting in more comprehensible and maintainable trans-

formations. Model transformation languages are explicitly designed for this task and result in better understandable and maintainable transformation as they offer adequate language constructs for expressing these relations. Second, the simulation-based metamodels provide a more abstract representation of the data model, simplifying understanding and abstracting from implementation details. Consequently, technical changes to a freight transport model that do not impact the data model are not propagated into the simulation-based metamodel and the model transformations and only affect the extraction and insertion components, leading to a more robust and maintainable architecture.

Subsequently, the single architecture components are discussed.

Common Metamodel

The common metamodel is a metamodel that encapsulates generally accepted and widely shared concepts within the freight transport domain. It aims to encompass all domain elements necessary at the exchange points defined in Section 4.2. This metamodel should exhibit a well-structured organization in terms of element dependencies, ensuring that elements included in a specific exchange point are independent of elements from other exchange points. Additionally, the common metamodel must have the capability to capture variability within the domain concepts, as discussed in Section 4.3.

From a technical perspective, a metamodel can be implemented as an Ecore model within the EMF framework [57]. Realizing the metamodel in different technical spaces is also feasible, provided that the common metamodel and simulation-based metamodels are realized within the same or compatible technical space to ensure interoperability. Moreover, the chosen technical space should provide suitable tools for creating, transforming, and managing metamodels effectively.

Simulation-Based Metamodel

Simulation-based metamodels refer to metamodels that depict the domain model of a particular freight transport model. These metamodels capture the unique characteristics of the specific freight transport model, facilitating straightforward extraction and insertion processes. However, they should abstract from technical implementation details and may exclude parts of the domain model not included in the utilized exchange points. Examples of simulation-based metamodels include the domain models developed for logiTopp (see Section 3.1) and MATSim-Freight (see Section 3.2).

Model Transformations

Exogen model-to-model transformations are used to transfer data between the simulation-based metamodel and the common metamodel, as well as vice versa. Given the potential variance in the level of abstraction and the information contained within these metamodels, the transformations may need to perform tasks such as aggregating or disaggregating data, ignoring certain information, or generating default values for missing elements. Transformations can employ imperative or declarative approaches, depending on factors such as task complexity and developer preferences.

Extraction and Insertion

The extraction and insertion components are tasked with creating an instance of a simulation-based metamodel or integrating it back into the freight transport model. Their implementation heavily relies on the provided interfaces, implementation, and input/output data of the freight transport model. Consequently, various approaches exist and must be selected based on the specifics of the existing freight transport model implementation.

Dynamic approaches involve extending or integrating into the code of the freight transport model. They collect data during runtime and programmatically generate a simulation-based model or insert it into the freight transport model dynamically. File-based approaches utilize model-to-text, text-to-model, or model-to-model transformations to create simulation-based models from file-based input/output data or vice versa. Additionally, hybrid and other strategies may be employed to achieve model extraction and insertion.

4.2 Exchange Points

Exchange points are points in the model process where data exchange and coupling between different models is possible. This section develops the exchange points and the underlying model structure used in the presented concept. We begin by defining criteria for selecting exchange points and then present the used exchange points along with the underlying model structure used in the presented concept (Section 4.2.1). In Section 4.2.2 then, possible use cases are presented briefly. We also present possible extensions to this structure in Section 4.2.3.

4.2.1 Model structure and Exchange Points

In Section 2.3.1, we presented a generalized model structure applicable to freight transport models. Furthermore, a discussion on the model structure of existing freight transport models, specifically logiTopp and MATSim-Freight, was presented in Section 2.3.2. We now define exchange points and a model structure used in the presented concept. We require the model structure to be compliant with these presented model structures. The exchange points and the use cases derived from them have to be defined in such a way, that the user can make use of the variety of functions of the models through the coupling of the models through these exchange points. So that the coupling produces a real added value.

We have formulated the following criteria for the selection of exchange points, which are presented and justified subsequently:

- **Natural separation**: Exchange points should naturally, in the sense of how the investigated models are structured, separate the model processes. The modeling steps between the exchange points should yield problems that are as separate and isolated as possible. This separation allows for the utilization of various models, algorithms, and concepts in individual modeling steps. Implementing a modeling step should afford maximum degrees of freedom, minimizing constraints and assumptions. Consequently, exchange points serve as input and output values for separated problems, enabling the interchange of solution algorithms or strategies without internal data exchange during problem-solving.
- **Thin exchange points**: A limited quantity and complexity of transferred data should characterize exchange points. This criterion seeks to streamline the common metamodel and required transformations, enhancing the manageability, comprehensibility, and maintainability of the system. Additionally, the performance of the implementation is a further aspect.
- **Common view existent:** The concepts employed by freight transport models at exchange points should align as closely as possible. There should be a common understanding of the concepts and data to be transferred. This criterion again aims to reduce the variability and resultant complexity, as differing concepts require handling of the introduced variability.

Both the selected extension points and the derived model structure are shown in Figure 4.2 and detailed as follows:

- **P0: Population Synthesis**: Responsible for generating a population comprising private individuals, businesses, and the network of transport service providers, this step draws upon various amounts and types of input sources. Additionally, it generates a representation of the general transport network.

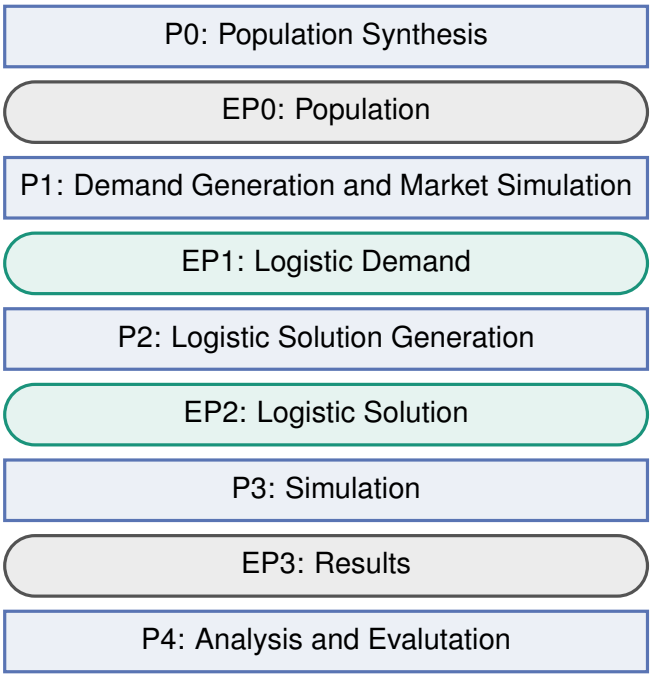


Figure 4.2: Model steps (P) and exchange points (EP). Gray exchange points are optional and not further elaborated in this work.

- **P1: Demand Generation and Market Simulation**: The objective of this model step is to deduce, based on a given population, the demand for transporting shipments and parcels. This entails specifying actual shipments at specific times, with assigned sizes, starting points, and destinations. Some models may treat this as an input parameter, while others employ a complex market simulation to derive and simulate business relations and supply chains, thereby determining the demand. Furthermore, this step involves simulating the transport market, encompassing the assignment of shipments to transport service providers and the relationships between them.
- **P2: Logistic Solution Generation**: The logistic solution generation tries to find an efficient or realistic logistic solution to handle the given demand. This involves the selection of transport chains, tour planning, and scheduling. Inputs for this phase primarily include the logistic demand and the logistic network.
- **P3: Simulation**: The simulation step is responsible for the execution of the evaluated logistic solution within the simulation environment. It entails a detailed simulation of logistic processes and interactions with other agents in the simulation environment, yielding a trace of the simulation (results), such as actual travel times or information about the success of deliveries.
- **P4: Analysis and Evaluation**: In this step, the captured simulation traces are reviewed, analyzed, and evaluated. Several tools could be applied to evaluate the simulation results, depending on the research questions.

At each extension point, the data exchanged encompasses the artifact derived from the preceding model step and the data from all prior extension points for reference. An extension point can be perceived as a problem definition for the subsequent model step. For instance, the logistic demand exchange point contains the logistic demand, logistic network, population, and transport network, defining a logistic solution problem where an optimal solution must be found for the given demand utilizing the available logistic network.

Note that carrier choice is distributed across two modeling steps, with market simulation responsible for simulating the transport market and selecting a carrier. However, logistic solution generation retains the flexibility to involve multiple carriers by allowing one carrier to commission one or more other transport service providers for a shipment, resulting in transport chains with multiple carriers involved.

Based on the evaluated models, the previously established criteria for selecting exchange points are largely met for EP1 and EP2. However, in the case of EP0 and EP3, there is a significant divergence in the common view. For EP0, the subsequent demand generation and market simulation step exhibit a wide range of strategies, varying from requiring demand as input data to employing statistical models and complex market simulations. Consequently, there are diverse requirements for the necessary input data, and the perceived added value of early model coupling is minimal for the user. A similar challenge arises with EP3, where the granularity and focus of the simulation step exhibit considerable variability. The subsequent analysis is again tailored to the main research questions of the particular model and, at least conceptually, coupled very tightly to the simulation step. Hence, we have chosen not to conduct a detailed examination of these exchange points in this work and have identified them as a research question for future investigations. The relevance of the results exchange point becomes more pronounced when feedback loops are taken into account. This is discussed in Section 4.2.3.

Nevertheless, certain elements of the data exchanged in EP0 are essential as a foundation for subsequent steps and exchange points. The data exchanged, particularly representing the general transport network and the population, serves primarily as a reference point utilized in both the logistic demand and the logistic solution. Consequently, the corresponding metamodel elements can be designed to accommodate these specific requirements and establish a minimal common view of the employed concepts. The design of the representation of the logistic network can be predominantly shaped by the needs of the logistic solution generation and simulation steps, coupled with the shared understanding of the domain.

It is not absolutely necessary for the sequence of the coupled models to fit exactly into this structure. In particular, the dynamic implementation of the extraction and insertion components allows data to be collected or fed back into the target model distributed

across several process steps.

4.2.2 Supported Use Cases

The model structure presented, with the utilized exchange points EP1 (logistic demand) and EP2 (logistic solution), offers a wide array of use cases for coupling freight transport models. To illustrate this versatility, Figure 4.3 showcases four potential use cases when coupling up to three freight transport models. These use cases are briefly described as follows:

- Red: Model A utilizes the logistic solution generation strategy of Model B.
- Purple: Model A conducts population synthesis, demand generation, and market simulation. Subsequently, Model B generates a logistic solution for the determined demand, which is then simulated, analyzed, and evaluated with Model C.
- Cyan: Model A generates both demand and a corresponding logistic solution. The logistic solution is then simulated, analyzed, and evaluated with both Model A and Model B.
- Orange: Given a logistic demand generated by Model C, two logistic solutions are produced using the strategies of Model B and Model C. These solutions are then simulated, analyzed, and evaluated with Model C. With the overall goal of comparing different different logistic solution strategies of Model B and Model C.

These use cases highlight the diverse ways in which freight transport models can be coupled to leverage their respective strengths and capabilities in addressing various aspects of the transportation domain.

4.2.3 Extensions: Feedback Loops and Model Merging

In this section, we briefly present two potential extensions to the previously outlined structure, which broaden the scope of use cases for model coupling. These extensions include feedback loops and model coupling. It's important to note that these extensions are not further elaborated in this work and are, therefore, left as areas for future research and exploration.

Feedback loops and iterative approaches are widely used concepts in freight transport models. They allow previous model steps to recompute their results based on the results of later model steps and therefore help to produce more accurate or optimized results. An example of the use of a feedback loop is the MATSim process (see Figure 2.11), where the agents optimize their plans based on the simulation results and scores of previous iterations.

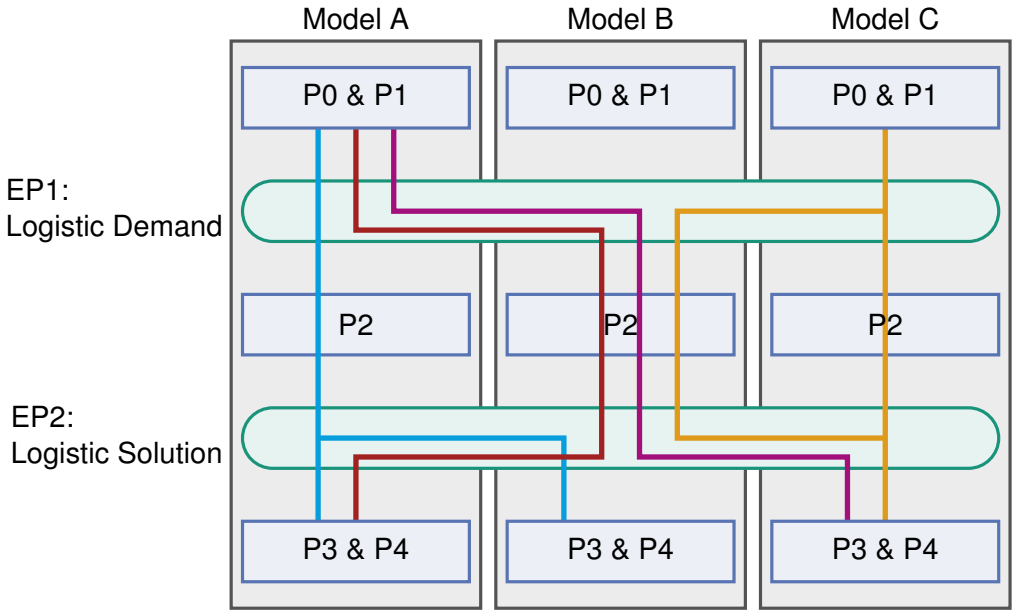


Figure 4.3: Example of various supported use cases with three coupled freight transport models.

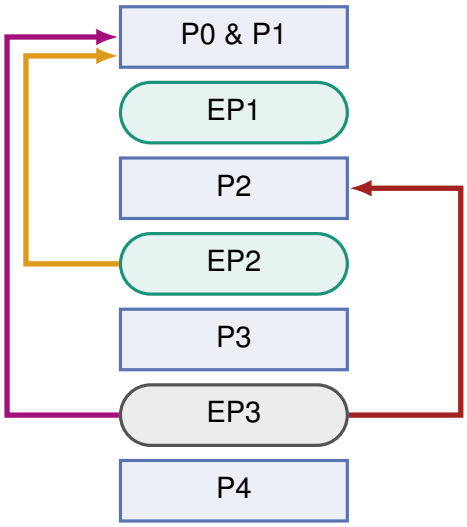


Figure 4.4: Possible feedback loops within the proposed model structure.

Figure 4.4 illustrates potential feedback loops within the proposed model structure. These are briefly outlined as follows:

- Red: This feedback loop returns simulation results to the logistic solution generation step. Here, the logistic solution generation could benefit from information such as experienced travel times and delivery success rates.
- Purple: The magenta feedback loop feeds simulation results back to the demand generation and market simulation step. This could include data on experienced travel times or pricing information for logistic services, aiding in more accurate demand prediction and market simulation.
- Orange: The orange feedback loop returns the generated logistic solution to the demand generation and market simulation stage. Here, the demand generation and market simulation could benefit from insights into the effort, costs, and service quality (e.g., transport times) associated with the found logistic solution.

Further research is required to delve into the specific data beneficial and necessary for these feedback loops. Additionally, the common domain model must be extended to include the required information. It must also be determined whether this data can be derived from existing metamodel representations through the application of model transformations.

A further extension involves merging metamodels at extension points, where multiple instances of a common metamodel are merged into a single instance. Figure 4.5 illustrates a potential use case where the generated demand from Model A and Model B are merged at the logistic demand exchange point. For instance, Model A might derive demand for private shipments, while Model B focuses on demand for business shipments. Model-driven engineering offers various tools for metamodel merging, with the Epsilon Merging Language (EML) being a popular example [29]. However, further exploration is needed to identify specific use cases for model merging in the domain of freight transport models. Additionally, it's important to address challenges such as

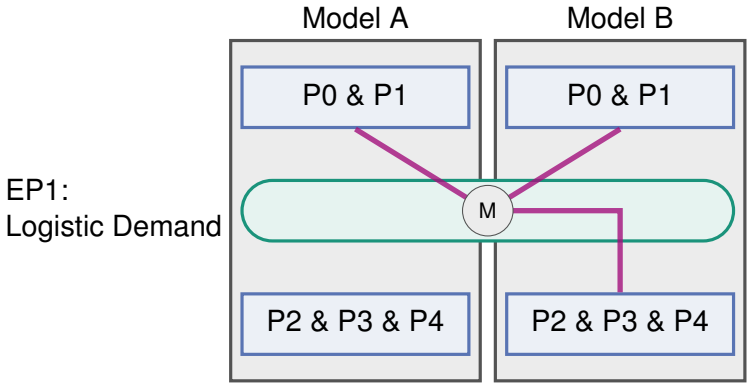


Figure 4.5: Example for merging the logistic demand generated by two models.

detecting and handling duplicated data elements to ensure the integrity and consistency of the merged metamodel.

4.3 Handling Variability

The common metamodel contains generally accepted and widely used concepts in the field of freight transportation. However, there are differences between the concepts in the analyzed domain models, which poses a challenge to compatibility. Consequently, the developed concept and the common metamodel need to address these differences and provide some kind of variability, which is discussed in this section.

For the description and handling of variability, we applied and adapted concepts from the field of Software Product Line Engineering (SPLE). Some fundamental and used concepts of SPLE are briefly described in Section 2.2.

4.3.1 Development Process

To address variability, we applied an adapted version of the engineering process for software product lines described by Apel et al. [6]. Instead of developing a specific product to meet customer needs in the application engineering part, we integrate freight transport models into the developed system.

In Chapter 3, we conducted a significant portion of the domain and requirements analysis. Following initial observations, commonalities and variability among metamodels are identified and structured into variation points and variants. The results of this analysis are presented in Chapter 5 as a feature model.

The domain implementation involves creating the common metamodel, reflecting the variation points and variants. Ideas for representing such variability are discussed in Section 4.3.3. The developed metamodel is presented in Chapter 5.

When adding a new freight transport model to the system, a requirement analysis is conducted, potentially leading to an adaptation of the common metamodel. The product derivation step includes developing model transformations between the new simulation-based model and the common metamodel. Additionally, the development of extraction and insertion components may be necessary.

Product derivation also differs in another aspect: Instead of deriving a product using the required fragments of the domain implementation, we require the instance of the common metamodel to fulfill the constraints implied by the required feature configuration of the metamodel. Therefore, we need further endogen model transformations on the

common metamodel to transform the metamodel between different variants. This process is further described in Section 4.3.4.

Note that the number of variation points should be kept small, and the choice of variation points should be made carefully. Since each variation point potentially adds additional complexity to the common metamodel and the transformations between the variation points. Thus, it may also be a valid decision not to map a concept of a freight transport model completely to avoid a variation point.

4.3.2 Allowed Variabilty

To represent and manage the variability of the common metamodel, we utilize feature models. To simplify our concept, we focus on a limited subset of feature models. Our approach involves a root node with three types of children:

- A concrete feature without further children, representing concepts shared among all models.
- An optional concrete feature without further children, representing concepts present in some freight transport models. This is referred to as an *optional variation point*.
- An abstract, mandatory feature, followed by an alternative group of concrete features with no further children. This is used to model variability in concepts representing the same thing but in different ways. This is referred to as an *alternative variation point*.

We leave it to further research to explore more complex feature models and how they could be integrated into our proposed concept.

4.3.3 Realization of Variability in the Common Metamodel

In this section, we briefly present some mechanisms for expressing variation points in the common metamodel. While a wide array of approaches, strategies, and patterns exist for this task, we limit our discussion to the mechanisms used in the developed metamodel in Chapter 5.

One approach is to use an abstract class to represent a variation point and have concrete subclasses for each variant. Another approach involves using specialized subclasses to enrich a metamodel element with the information required for a variation point. Optional variation points can also be realized as additional metamodel elements, with careful consideration to ensure that non-optional elements do not depend on optional ones.

A further strategy is to employ flexible model structures that can represent multiple variants. The properties of a specific variant can then be described by formal model constraints.

In general, formal model constraints can be used to specify the invariants of each variant and to verify if the variant is fulfilled.

4.3.4 Realization of Variability in the Model Coupling

This section outlines the handling of a variable common metamodel in the proposed concept.

A feature configuration is derived for each simulation-based metamodel, representing its utilized concepts. This configuration ensures conceptual alignment between the simulation-based metamodel and the common metamodel. The transformation from the simulation-based metamodel to the common metamodel then produces an instance of the common metamodel that is consistent with this feature configuration. The opposite transformation can assume that the instance of the common metamodel is again consistent with this feature configuration. This idea of transformations producing and requiring an instance of the common metamodel with a specific feature configuration is shown in Figure 4.6.

When coupling two freight transport models, such as Model A and Model B, the transferred instances of the common metamodel need to be transformed from an instance that fulfills the produced feature configuration of Model A to an instance that fulfills the required feature configuration of Model B. Thus, a set of endogen model transformations on the common metamodel, further referred to as common transformations, must be defined.

Common transformations cater to different types of variation points:

- Optional variation points require two transformations: one for removing optional concepts from the metamodel and another for generating and adding the elements of the optional concept.
- Alternative variation points necessitate transformations to switch between each alternative. Chaining transformations might be applicable in some cases. For example, instead of developing a transformation from $A \rightarrow B$, the transformations $A \rightarrow C$ and $C \rightarrow B$ could be chained to $A \rightarrow C \rightarrow B$.

It's important to note that not all the information required to create specific concepts may be derivable from the source of a common transformation. Strategies like dummy data creation or user input may be employed in such cases. It also must be noted that these transformations could potentially lead to a loss of information, as not all variants

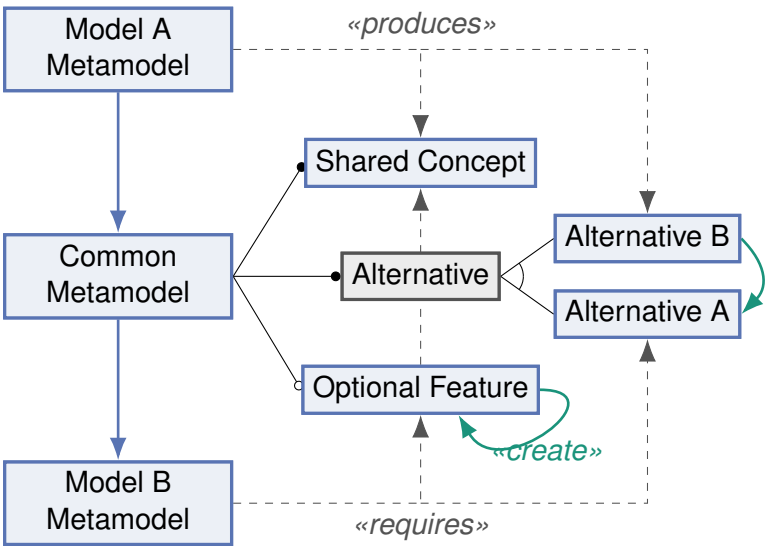


Figure 4.6: Example for the *produced* and *required* feature configuration of the common metamodel when coupling two simulation-based metamodels and the required common transformations (green).

of a variation point contain the same information.

Common transformations should ideally be independent of the configuration of other variation points and capable of handling various configurations. However, constraints on some features might be beneficial in certain scenarios.

Common transformations must be defined in such a way that the resulting models exhibit only minor differences in content, which are not considered significant, regardless of the order in which the set of transformations required to map between two feature configurations is applied. However, in this work, we do not delve further into examining if this property can hold in all circumstances and which kinds of differences are permissible. If this property cannot be guaranteed, further investigation is needed to determine how to establish correct execution orders. We leave the exploration of both aspects, aided by formal methods, to future work.

We identify three key benefits in the proposed approach of introducing feature configurations for every coupled freight transport model and integrating common transformations to facilitate the mapping between them:

- Simplified Transformations: The transformations between simulation-based and common metamodels are expected to become less complex. This is because the concepts of the source and destination metamodels are aligned through the application of a feature configuration that reflects the concepts of the simulation-based metamodel.
- Shared Transformations: Many parts of the transformations that would be required to map between a simulation-based metamodel and the common metamodel

without variability are now shared as common transformations. This consolidation reduces the effort needed for development and maintenance.

- Reduced Information Loss: By avoiding the transfer of all simulation-based metamodels into a single common metamodel with no variability, the loss of information can often be minimized. The common metamodel can incorporate shared concepts, and if both the source and destination simulation-based models include the concept, the loss of information is minimal. Additionally, specialized transformations can be developed to map between variants of a variation point rather than initially constraining the information to a fixed concept of the common metamodel.

5 Common Freight Transport Metamodel

This chapter introduces the common metamodel developed for the freight transport domain, serving as a central component for coupling various simulation-based metamodels (refer to Chapter 4). We commence by outlining the feature model that delineates the variability of the common metamodel. Subsequently, we delineate the elements of the metamodel organized by several view types, analogously to the presentation of the metamodels of logiTopp and MATSim-Freight in Chapter 3. Thereby, we briefly deliberate on the associated design decisions and offer insight into how the concepts of the common metamodel can be aligned with the examined simulation-based metamodels and vice versa. Finally, in Section 5.8, we elaborate on the implementation of transformations for changing variants at the variation points.

The feature model in Figure 5.1 illustrates the variability of the common metamodel through various variation points and their respective variants. This is the result of a requirements and domain analysis and includes:

- Shared Concepts: This feature is included for the sake of completeness and represents all non-variable and, therefore, shared concepts/features of the metamodel.
- Shipment Records: An optional feature capturing an explicit representation of shipment records. Although redundant, shipment records serve algorithmic purposes and can be derived from planned or executed tours. The reason for making this feature optional is to provide a common transformation for creating this structure and thereby share the code and knowledge with all transformations from simulation-based metamodels to the common metamodel.
- Dimension: Describing how capacities and sizes are depicted within the metamodel, offering volume-based, weight-based, or combined representations.
- Network Access: Describes how the position of locations within a network is described. Locations can either be connected to a node or an edge of the network.
- Simulation Period: This variation point describes the length of the simulation period. It can represent a single day or multiple days (currently constrained to seven days).
- Transport Chains: Determining whether complex transport chains (including

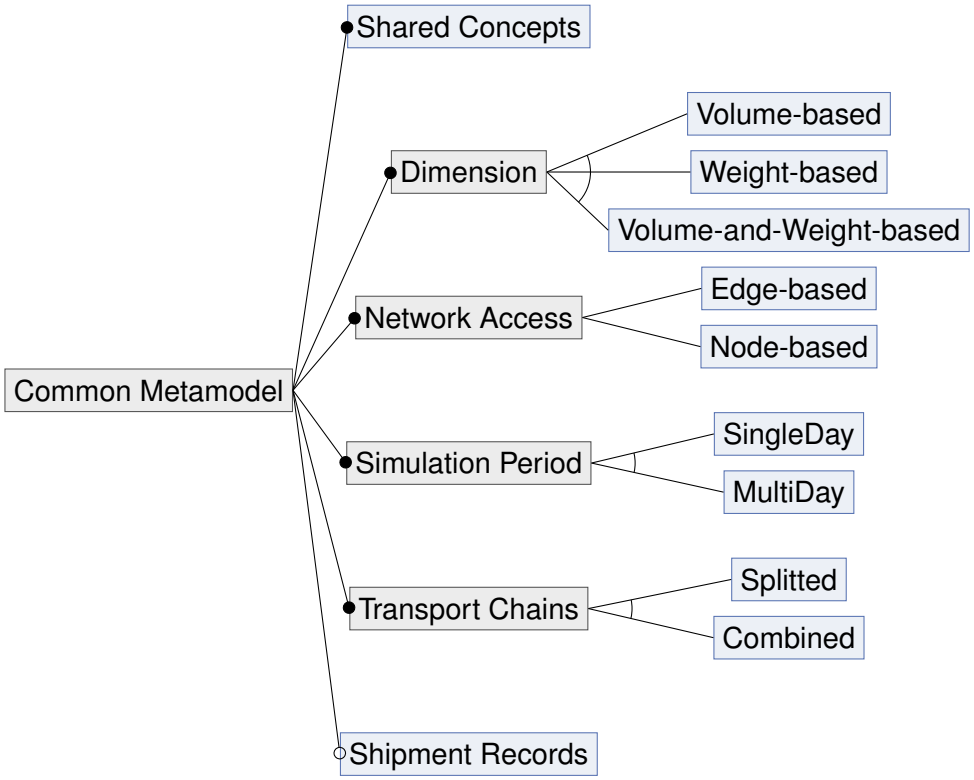


Figure 5.1: Feature diagram of the common metamodel.

multiple hops) are represented as a single entity or split into multiple shipments, each representing one hop.

The feature configuration produced respectively required for the transformations from and to the logiTopp metamodel is {*Shared Concepts*, *Volume-based*, *Edge-based*, *MultiDay*, *Combined*}. Analogously, for the MATSim-Freight metamodel, the feature configuration is {*Shared Concepts*, *Volume-based*, *Edge-based*, *SingleDay*, *Splitted*}. Note that MATSim-Freight would also allow weight-based dimensions.

The common metamodel, developed using EMF's Ecore metamodel [57], is structured based on viewpoints defined in Chapter 3, with each viewpoint corresponding to a package within the metamodel. The packages were assembled so that the packages required in an exchange point depend only on packages from previous exchange points. The resulting common metamodel contains the following set of packages $P = \{Utils, Network, Population, Logistic\ Network, Logistc\ Demand, Logistic\ Solution, Results\}$. The package dependencies are listed below, in your notation $A \rightarrow \{B,C\}$ denotes that Package A depends on both Packages B and C:

- $Utils \rightarrow \{\}$
- $Network \rightarrow \{Utils\}$
- $Population \rightarrow \{Utils, Network\}$

- $Logistic\ Network \rightarrow \{Utils, Network\}$
- $Logistic\ Demand \rightarrow \{Utils, Network, Population, Logistic\ Network\}$
- $Logistic\ Solution \rightarrow \{Utils, Network, Logistic\ Network, Logistic\ Demand\}$
- $Results \rightarrow \{Utils, Network, Logistic\ Network, Logistic\ Solution\}$

Note that certain metamodel elements may appear in multiple viewpoints, or they might be introduced at points where they are semantically used rather than within their contained viewpoint. This approach aids clarity and ensures that elements are presented in context, enhancing understanding for the reader. We commence now by presenting the resulting view types.

5.1 Utils View Type

The metamodel has a further utils view type and package that encapsulates utility or helper elements and functionalities that support the main modeling concepts within the metamodel. The utils view type encompasses elements for describing times and dimensions in terms of size and capacities. Therefore, it includes major parts of the realization of dimension and simulation period variation points.

Figure 5.2 shows the metamodel elements related to time, time windows, and durations. Subsequently, we briefly describe these elements:

- A *Duration* represents a specific period of time and includes the fields *days*, *hours*, *minutes*, and *seconds* to define its duration precisely.
- The abstract classes *Timestamp* and *TimeWindow* are used to implement the simulation period variation point. If the feature *MultiDay* is selected, only *MultiDay-Timestamps* and *MultiDayTimeWindows* will be allowed. If the feature *SingleDay* is selected, only *SingleDayTimestamps* and *SingleDayTimeWindows* will be allowed.
- A *(Single/Multi)DayTimestamp* is used to specify a point in time during the simulation period and includes the attributes (*day*,) *hour*, *minute*, and *second* to define the timestamp precisely.
- A *(Single/Multi)DayTimeWindow* is used to specify a time window. It starts at a timestamp (*from*) and ends at a timestamp (*to*). Both *from* and *to* are optional to express that the start or end of the *TimeWindow* is not explicitly defined.

We decided to use an explicit representation for times to enhance readability for the user. The mapping to the logiTopp and MATSim-Freight metamodel is straightforward. The precision in seconds is sufficient for most use cases, and the proposed structure is very flexible and allows for the representation of all kinds of timestamps, time windows, and durations. If necessary, the flexibility could be reduced through the constraints of

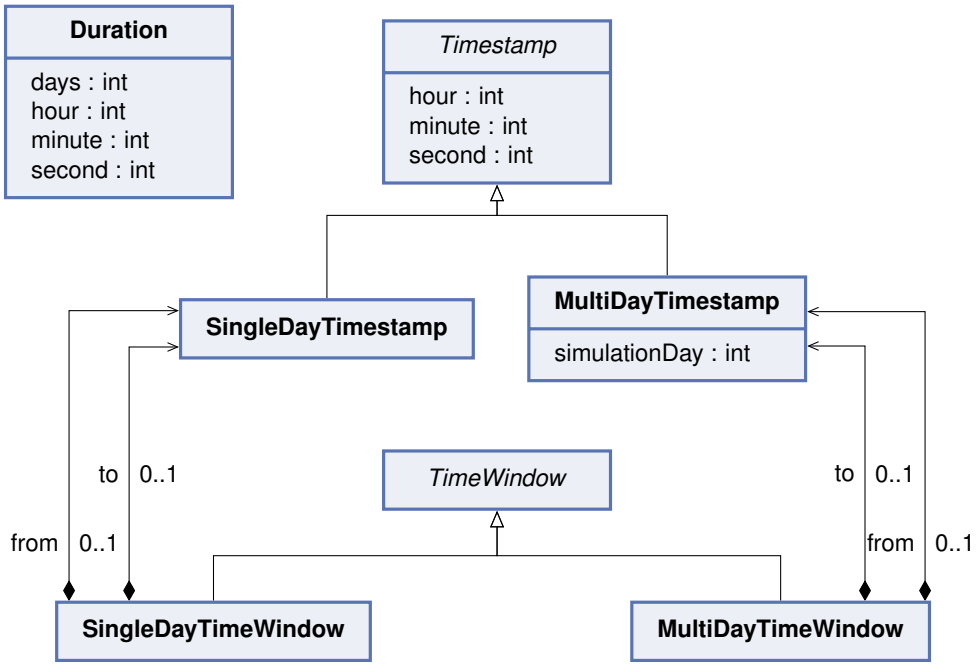


Figure 5.2: Common Metamodel: Utils View Type - Times, Timestamps and Durations.

metamodel elements that use these elements.

Figure 5.3 shows the metamodel elements related to the representation of dimensions.

- The abstract class *Dimension* implements the same named variation point. It allows either a volume-based, a weight-based, or a combined (*VolumeAndWeightBasedDimension*) representation.
- *VolumeBasedDimensions* can either have a finite value in m^3 or be infinite.
- *WeightBasedDimension* can either have a finite value in kg or be infinite.
- *VolumeAndWeightBasedDimensions* have both a volume and a weight. The semantics for determining if a size s can be handled by a capacity c are to check both volume and weight. The size s then can be handled by the capacity if the following holds: $s.weight \leq c.weight$ and $s.volume \leq s.volume$.

Volume and weight seem to be the two decisive quantities for describing sizes and capacities. Mapping between simulation-based models and dimensions is straightforward, as only units may have to be converted.

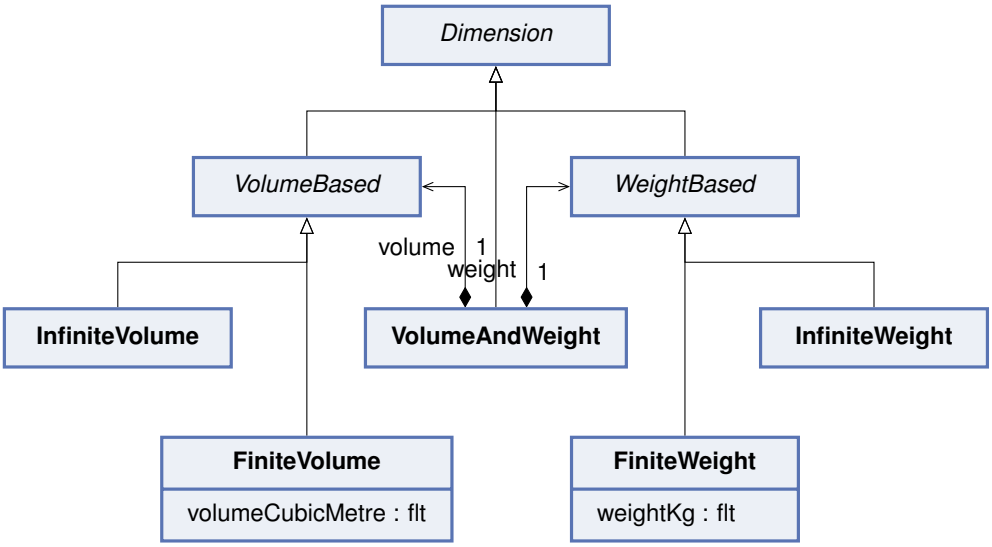


Figure 5.3: Common Metamodel: Utils View Type - Dimensions.

5.2 Network View Type

Before introducing the network view type of the common metamodel, we motivate three major design decisions that were made to create this view type:

- Level of detail of the network: Although basic concepts of most metamodels align (i.e., graph-based representation), the level of detail and the present properties of the network metamodels vary significantly. Deriving a common subset poses challenges. To address this, we opted for a simplistic network representation, capturing fundamental structure and entities. Each network element is assigned a unique id, facilitating mapping to more complex representations that can be transferred separately or are derived from the same source and, therefore, have matching ids.
- Representation of locations: Locations in networks can be represented diversely, often including coordinates and references to network elements through which the network is reached. To accommodate this variability, we introduced a variation point for location representation, acknowledging the different approaches available. A heuristic transformation between edge-based and node-based network accesses is possible.
- Containment of locations: In terms of location containment, we opted for instance equality over data equality for clarity in modeling. Locations are uniquely identified and belong to the containment structure of the entity they serve. Tours and similar elements then reference these locations, ensuring a clear delineation of relationships.

Figure 5.4 shows the elements representing the directed network graph. A network

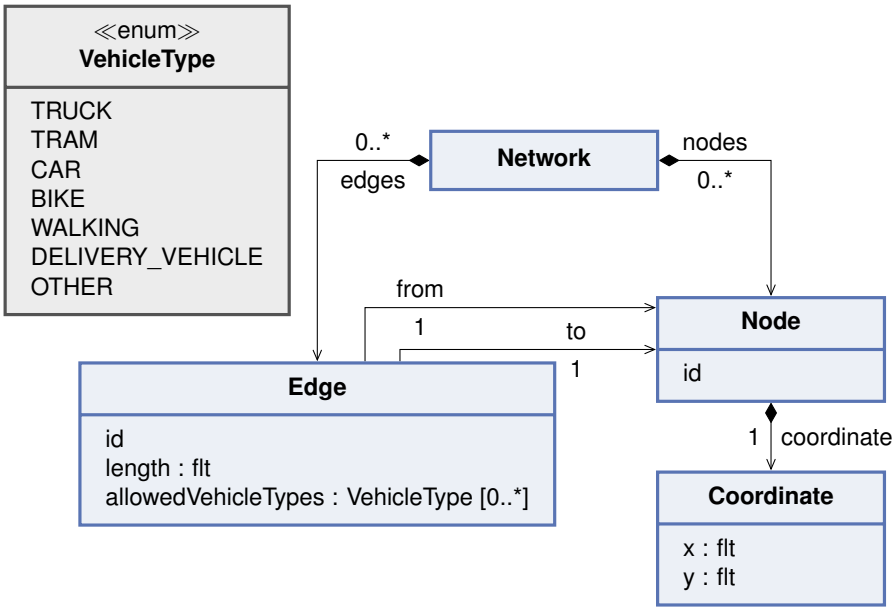


Figure 5.4: Common Metamodel: Network View Type - Network graph.

is composed of *Nodes* and directed *Edges*, each having an *id*. *Edges* further have a set of *allowedVehicleTypes* and a *length*. The unit for lengths and the specification of coordinates is set to meters. The *VehicleTypes* enum encompasses a set of typical used vehicle types in the CEP domain, and a mapping to the vehicle types of a simulation-based metamodel must be established. Some simulation-based metamodels contain zones, which are mainly used in the domain to define the areas of responsibility of logistic facilities. We have decided not to model zones in the common metamodel, as we see it as a degree of freedom for the planning algorithms to define the areas of responsibility of logistic facilities.

The metamodel elements for describing a location are shown in Figure 5.5. *Locations*

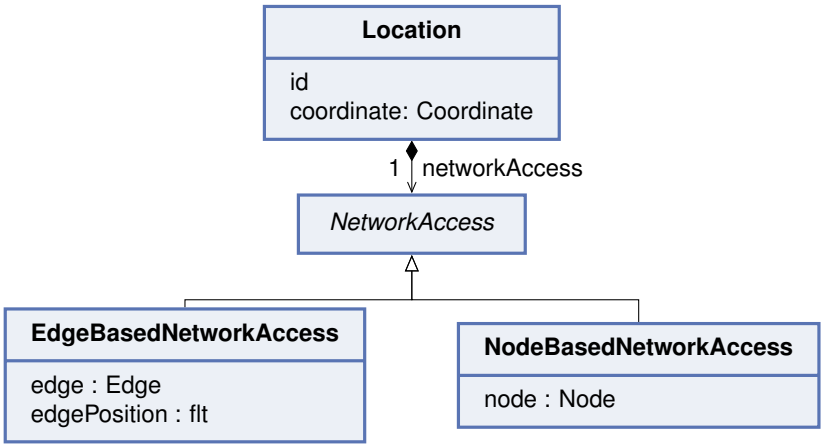


Figure 5.5: Common Metamodel: Network View Type - Locations.

are characterized by an *id* and a *coordinate*. Additionally, they have a *networkAccess*, which is an abstract class. Depending on the selected variant, this *NetworkAccess* can take two forms: an *EdgeBasedNetworkAccess*, which points to an *edge* and includes a relative position (ranging from 0 to 1) on the *edge* denoted as *edgePosition*, or a *NodeBasedNetworkAccess*, which points to a *node*. If no *edgePosition* is specified, a default value of 0.5 can be assigned.

Transformations from metamodels that do not contain an instance-based equality mechanism to the common metamodel can be challenging. This is because multiple locations of the source metamodel have to be merged into one location, and the container of the location has to be determined. A suitable strategy is to build a lookup table that assigns all locations of the source metamodel to the used instance. The lookup table is built upon an equality comparison of attributes with some tolerance.

5.3 Population View Type

Also, for the population view type, the level of detail had to be decided. We again opted for a simple and reduced representation as we excluded the EP0 for the exchange population (see Section 4.2.1). Therefore, the population is mainly used as a reference for describing producer and consumer relationships in the demand view type and does not need to contain attributes to derive the logistic demand.

Figure 5.6 shows the metamodel elements related to the population view type. Subsequently, we briefly describe these elements:

- The *Population* encompasses *businesses* and *households*.
- *Households* are characterized by an *id*, a *location*, and multiple *members*. Each *Person* has a unique *id* and belongs to exactly one household.
- We decided to include a distinction between businesses and branches. *Businesses* can have multiple *branches* and serve as a structuring concept. Each *BusinessBranch* has its own *id*, a specific *location*, and *openingHours*, which specify when a CEPSP can deliver or pick up parcels at the branch.
- Additionally, we added the role of a *ShipmentConsumerProducer* as an abstract class, indicating whether an entity can send or receive shipments.

This structure aligns with the examined metamodels, although some may not distinguish between businesses and branches. In such cases, a business with a single branch is mapped to a business containing one branch. Furthermore, a default value must be set if certain metamodels do not include opening hours for businesses/branches.

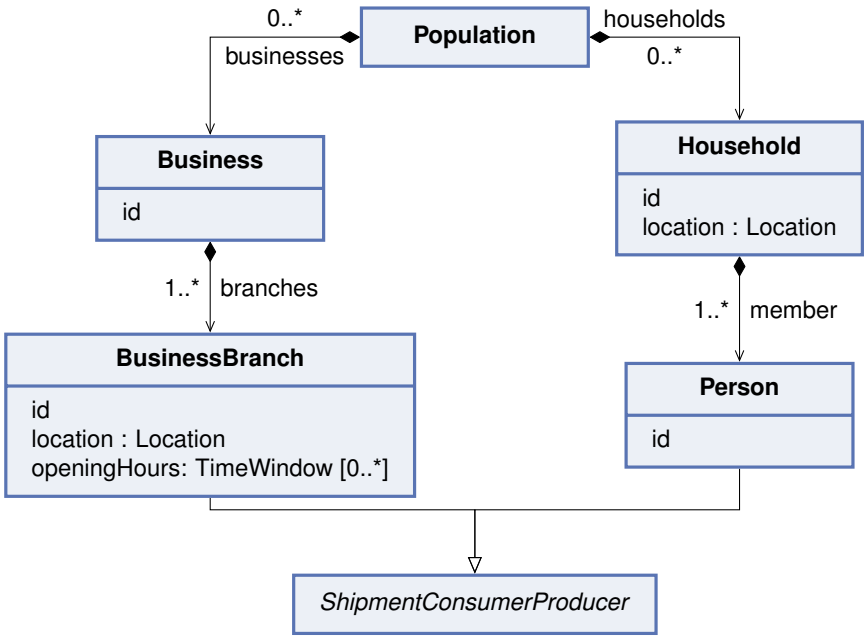


Figure 5.6: Common Metamodel: Population View Type.

5.4 Logistic Network View Type

The logistic network view type encompasses all metamodel elements used to describe the network and capabilities of CEPSPs. Before describing the elements of the logistic network view type, we motivate some major design decisions that were made to create this view type:

- Relations between CEPSPs only on the CEPSP level: We opted to model relations between CEPSPs, which is crucial for defining transport chains, solely at the CEPSP level for flexibility. This allows planning algorithms to determine useful transfers between hubs, considering contractual relationships between CEPSPs rather than prescribing allowed transport chains. Modelers or transformations can also introduce more CEPSPs to restrict the set of permitted transport chains, ensuring a flexible structure.
- Separation of logistic hubs and vehicle depots: We clearly distinguished logistic hubs and vehicle depots in the common metamodel, as they serve different purposes. This decision prevents complexities in mapping vehicles not located at logistic hubs, as this distinction is already present in some freight transport models.
- No representation of timetables: Timetables were omitted from the common metamodel due to their absence in some metamodels and varying conceptualizations across models. We leave the representation of timetables as a detail specific to the network representation of individual models.

- Explicit representation of public service points: We explicitly represented public service points in the common metamodel, considering it a central concept in the CEP domain. The attributes associated with public service points are deemed important constraints for deriving logistic solutions.

Figure 5.7 shows the metamodel elements related to describing CEPSPs. Subsequently, we briefly describe these elements:

- The *LogisticNetwork* serves as the root element of the logistic network and contains a set of *CEPSPs*.
- A *CEPSP* has a *name* and an *id*. Every CEPSP can operate multiple *vehicleDepots* and *logisticHubs*. Additionally, CEPSPs can cooperate with other CEPSPs, denoted by the *deliveryPartners* and *pickUpPartners* references.
- A *VehicleDepot* can have multiple *vehicles* and has a defined *location*. It also has an *id* and *operationHours*, denoting the time windows during which the vehicles can be operated.
- *Vehicles* have a unique *id*, a *type*, and *storageCapacity*. Additionally, a vehicle can only be operated during its *operationHours*, which must be within the *operationHours* of the containing *VehicleDepot*.
- A *LogisticHub* serves as a transshipment point of the logistic network. It has a limited *storageCapacity*, and shipments take at least the *minimumTranshipmentTime* to be processed in the LogisticHub. Furthermore, *LogisticHubs* serve as points where shipments can enter or exit the study area if the respective flags are set (*isEntry* and *isExit*). A *LogisticHub* has an *id*, a *location*, and *operationHours* during which vehicles can load and unload shipments.
- We further introduced the role of a *LogisticFacility* as an abstract class.

We now outline the logistic solutions allowed within this structure and the responsibilities of CEPSPs and logistic hubs. Shipments can be transported between logistic hubs or from a pickup or delivery location (e.g., a *PublicServicePoint*, *BusinessBranch*, or *Household*) to a logistic hub or vice versa. For transport between logistic hubs of the same CEPSP or the final delivery or pickup of shipments, the CEPSP that owns the respective logistic hub is responsible and must provide the vehicle used. For transport between logistic hubs of different CEPSPs, the CEPSP of the origin hub is responsible in delivery direction, and the CEPSP of the destination hub is responsible in pickup direction. These transports are only allowed if the corresponding delivery or pickup partner relation exists. In the case of a shipment with an origin and destination within the study area, the shipment is considered a pickup until it reaches a logistic hub of the responsible CEPSP, after which it is handled as a delivery. For this, a responsible CEPSP is assigned for every shipment in the demand view type.

The transformations for transferring the CEPSPs into the common metamodel can be

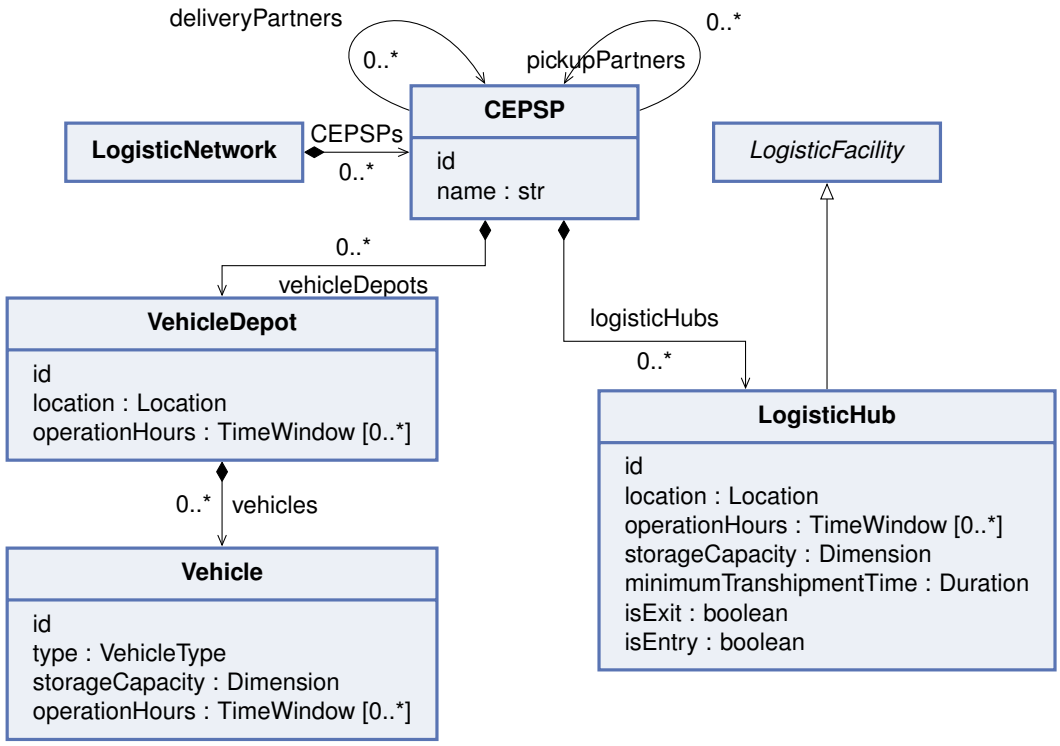


Figure 5.7: Common Metamodel: Logistic Network View Type - CEPSPs.

quite complex and may require manual intervention or adjustments. Conversely, when transforming from the common metamodel to other metamodels, it's often feasible to omit missing features or transfer properties to child or parent entities if concepts aren't modeled at the same level. Since not all metamodels include attributes like operation hours and minimum transshipment times, defining suitable default values or manually inputting operation hours may be necessary. Similarly, flags such as *isEntry* and *isExit* are often not explicitly modeled. A viable strategy for automatically deriving them is to analyze the provided logistic solution and set the flags to true if any shipment enters or leaves the study area at a respective LogisticHub. The same approaches can be employed for pickup and delivery partner relations.

The metamodel elements related to describing public service points are shown in Figure 5.8. Subsequently, we briefly describe these elements:

- The abstract class *PublicServicePoint* acts as a supertype for all public service points, possessing attributes such as an *id*, a *location*, and a limited *storageCapacity*.
- Concrete implementations of *PublicServicePoint* include *Packstations* and *Shops*. While shipments can be picked up and delivered at *Packstations* at any time, *Shops* have specified *openingHours* during which pickup and delivery are possi-

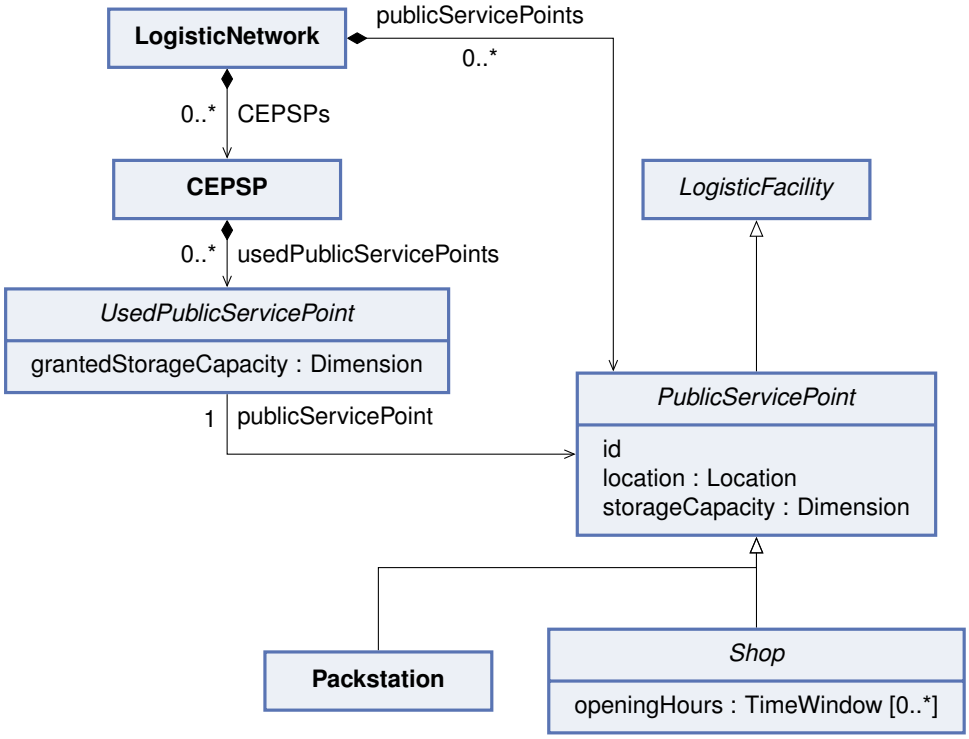


Figure 5.8: Common Metamodel: Logistic Network View Type - Public service points.

ble.

- *PublicServicePoints* are integral parts of the general *LogisticNetwork* and can be utilized by *CEPSPs*. This relationship is expressed through the *UsedPublicServicePoints* relation, which also indicates how much of the *storageCapacity* of a *PublicServicePoint* can be utilized by a *CEPSP*. The sum of the *grantedStorageCapacity* to all CEPSPs that use a specific *PublicServicePoint* must not exceed the *storageCapacity* of the *PublicServicePoint*.

In many metamodels, *UsedPublicServicePoints* are not explicitly modeled. However, a reasonable strategy to derive *UsedPublicServicePoints* is to analyze the demand, identify origins and destinations of shipments that are not locations of other receivers or logistic facilities, and then create a public service point at these locations. *UsedPublicServicePoints* can also be derived from these shipments. Additionally, reasonable default values for properties not explicitly contained must be defined.

While the network and population view types align well with other simulation-based metamodels and are primarily used for referencing, containing minimal semantics, the logistic network view type provides a framework for deriving logistic solutions, thus incorporating more sophisticated semantics. Consequently, metamodel concepts may diverge to a greater extent, increasing the complexity of transformations between them. To effectively implement these transformations, a comprehensive understanding of

the mapping and relationships among the source and target metamodels elements is required. This understanding is paramount as subsequent viewpoints, such as logistic demand and logistic solution, rely on these foundational concepts and often reference elements of the logistic network. Maintaining an explicit lookup table detailing mapped instances can greatly facilitate this process, aiding in the tracking and management of associations between elements across metamodels.

5.5 Logistic Demand View Type

The logistic demand view type essentially delineates the shipments that need to be transported by the CEPSPs. We categorize shipments based on whether their origin is inside or outside the study area and whether their destination is inside or outside the study area. This categorization results in four types of shipments. However, shipments with origins and destinations outside the study area are disregarded. Additionally, there is a requirement to provide an explicit representation for shipments that have been split or shortened due to the simulation period and transport chain variation points.

The following design decisions and considerations further influenced the modeling of this view type:

- No fixed origin/destination for entry or exit of the study area: We regard it as a strategic decision for the logistic solution to determine at which logistic hub a shipment should enter or leave the study area. Therefore, we do not prescribe fixed origin or destination locations in this scenario. Instead, the shipment can enter or exit the study area at any logistic hub of the responsible CEPSP where entry or exit is permitted.
- Distinction between destination/start location and sender/receiver: We opt to explicitly differentiate between the sender/receiver and the destination locations. This allows for dynamic changes to the destination in freight transport models while also providing fixed destinations to models that require them.
- No fixed service time: Some metamodels explicitly model the time required for pickup or delivery of a shipment, while others derive it dynamically based on factors such as recipient accessibility or the number of parcels being delivered concurrently. Therefore, we choose not to include this property in the common metamodel and delegate the derivation of these values to the models or transformations.

The logistic demand view type comprises a set of abstract base classes used to describe properties according to the previous categorization, from which concrete shipments are derived. These base classes, shown in Figure 5.9, are briefly described

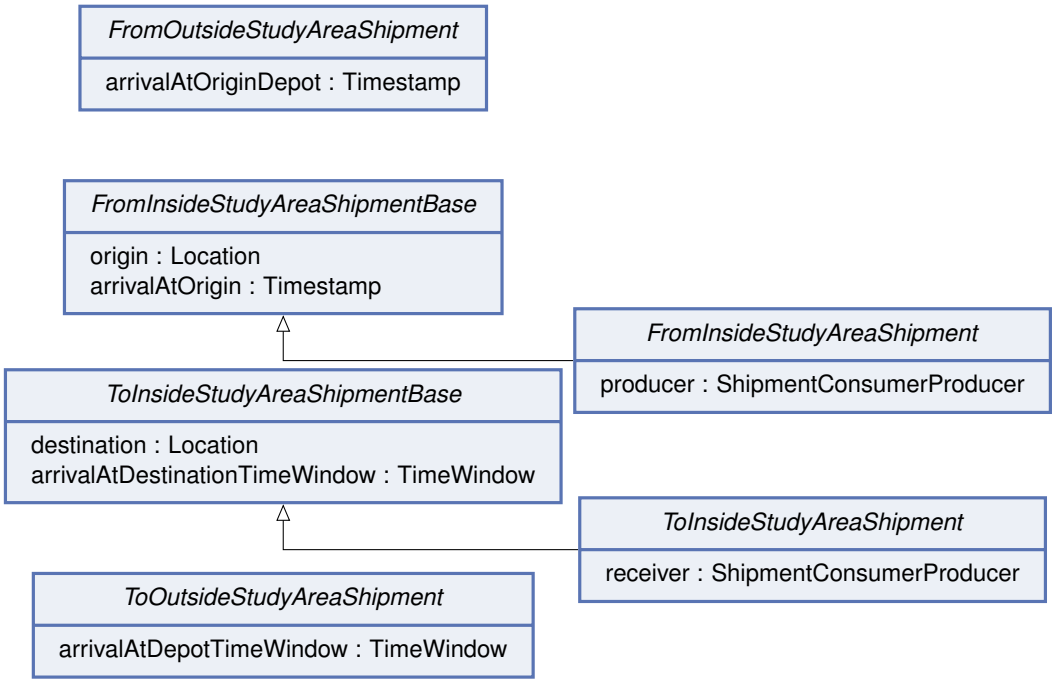


Figure 5.9: Common Metamodel: Logistic Demand View Type - Shipment base classes.

as follows:

- *FromOutsideStudyAreaShipment*: This shipment can arrive at any logistic hub of the responsible CEPSP that permits entry. The arrival time at this first logistic hub is specified by the *arrivalAtOriginDepot* property.
- *FromInsideStudyAreaShipmentBase*: This shipment must be picked up at its *origin* location at or after the specified *arrivalAtOrigin* time. The *FromInsideStudyAreaShipment* extends this base class and includes a reference to the *ShipmentConsumerProducer* that sends the shipment (*producer*).
- *ToOutsideStudyAreaShipment*: This shipment can exit the study area at any logistic hub of the responsible CEPSP that permits entry. Optionally, a time window can be specified (*arrivalAtDepotTimeWindow*) to describe when the shipment must leave the study area.
- *ToInsideStudyAreaShipmentBase*: This shipment must be delivered at its destination location. Optionally, a time window can be specified to describe when the shipment must be delivered (*arrivalAtDestinationTimeWindow*). The *ToInsideStudyAreaShipment* extends this base class and includes a reference to the ShipmentConsumerProducer that receives the shipment (*receiver*).

The concrete types of shipments are derived from a base class and the base classes of their respective category, as depicted in Figure 5.10. Each *Shipment* is characterized by an *id*, a *size*, and a *responsibleCEPSP*. There are four types of concrete shipments:

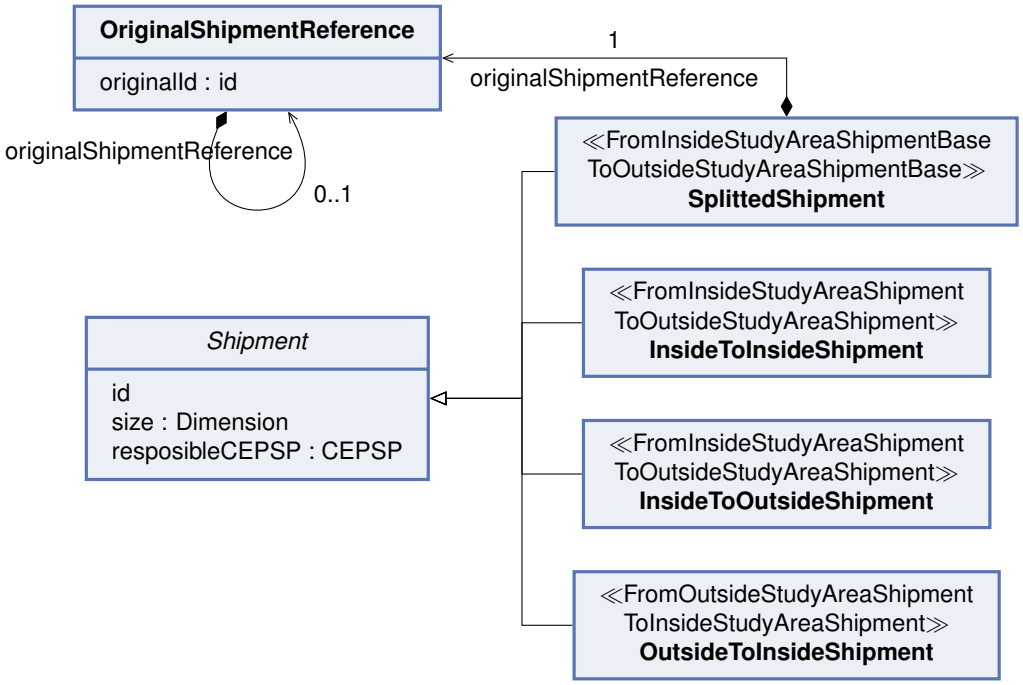


Figure 5.10: Common Metamodel: Logistic Demand View Type - Shipments.

- *InsideToInsideShipment*: This shipment is sent from a producer to a consumer within the study area.
- *OutsideToInsideShipment*: These shipments originate from outside the study area and are delivered within the study area.
- *InsideToOutsideShipment*: These shipments originate within the study area and are delivered outside of it.
- *SplittedShipment*: Used to describe a subset of a transport chain, a *SplittedShipment* has no explicit producer and consumer. It may start or end at a logistic hub that is not available for entry or exit of the study area. To maintain the connection with the original shipment, a *SplittedShipment* contains a reference (*originalShipmentReference*) to the original shipment's id (*originalId*). Note that a shipment can potentially be split multiple times, leading to nested references within the originalShipmentReference.

Transformation of logistic demand between the common metamodel and simulation-based metamodels and vice versa should be straightforward in most cases. A lookup of the population locations can determine producers and consumers if they are not explicitly modeled. A further challenge may be the determination of arrival time (windows), where it may be helpful to specify suitable default values.

5.6 Logistic Solution View Type

The logistic solution view type of the common metamodel comprises two separate but mostly redundant representations. The mandatory representation views the logistic solution as the tours planned by the CEPSPs. The second representation, which is an optional feature, represents the logistic solution in the form of shipment records. These shipment records point to the planned tours and can be derived from them. This redundancy was introduced to simplify tracking a shipment's transport chain for both the model user and the common transformations. Additionally, it facilitates modeling and tracking failed pickup and delivery attempts in the results view type with this second representation.

The metamodel elements describing the planned tours of a logistic solution are shown in Figure 5.11. Subsequently, we briefly describe these elements:

- The *LogisticSolution* serves as the root element and contains both a set of planned *tours* and, optionally, the *shipmentRecords*.
- The abstract class *Tour* describes any kind of tour. A *Tour* is executed by a *vehicle* and has an *id*. It further consists of a list of *stops*. A *PlannedTour* is a *Tour* that represents a planned tour, specifies the CEPSP that is executing the Tour (*executingCEPSP*), and thus also must provide the vehicle.
- A *Stop* is the abstract base class for several types of stops and includes an *id*, a stop number (*no*), and a time window during which the stop shall be executed (*stopTimeWindow*). Additionally, the *location* of the stop is specified. These properties enable the derivation of travel times and distances between stops, eliminating the need for explicit modeling.
- A *StopLocation* can be either a *ReferenceStopLocation* that references a *location* of the logistic network or the population or a *CustomStopLocation* that contains a new *location*. This distinction allows for creating stops at any point in the network, such as when multiple different receivers are served at once. The reference stop location also facilitates navigation from the stop location to the associated entity via opposite references or containment relations.
- The first and last stops of a tour must be *StartEndStops*, specifying at which vehicle *depot* a tour starts and ends. The stop location must be identical to the location of the *depot*.
- *PickupDeliveryStops* are stops at which parcels are picked up and delivered. They contain a set of shipments that shall be unloaded (*unloadedShipments*) and loaded (*loadedShipments*) at the stop. This type of stop can either take place at a logistic facility (*LogisticFacilityStop*), where again the locations of the logistic facility and the stop must be identical, or anywhere within the network (*NormalStop*).

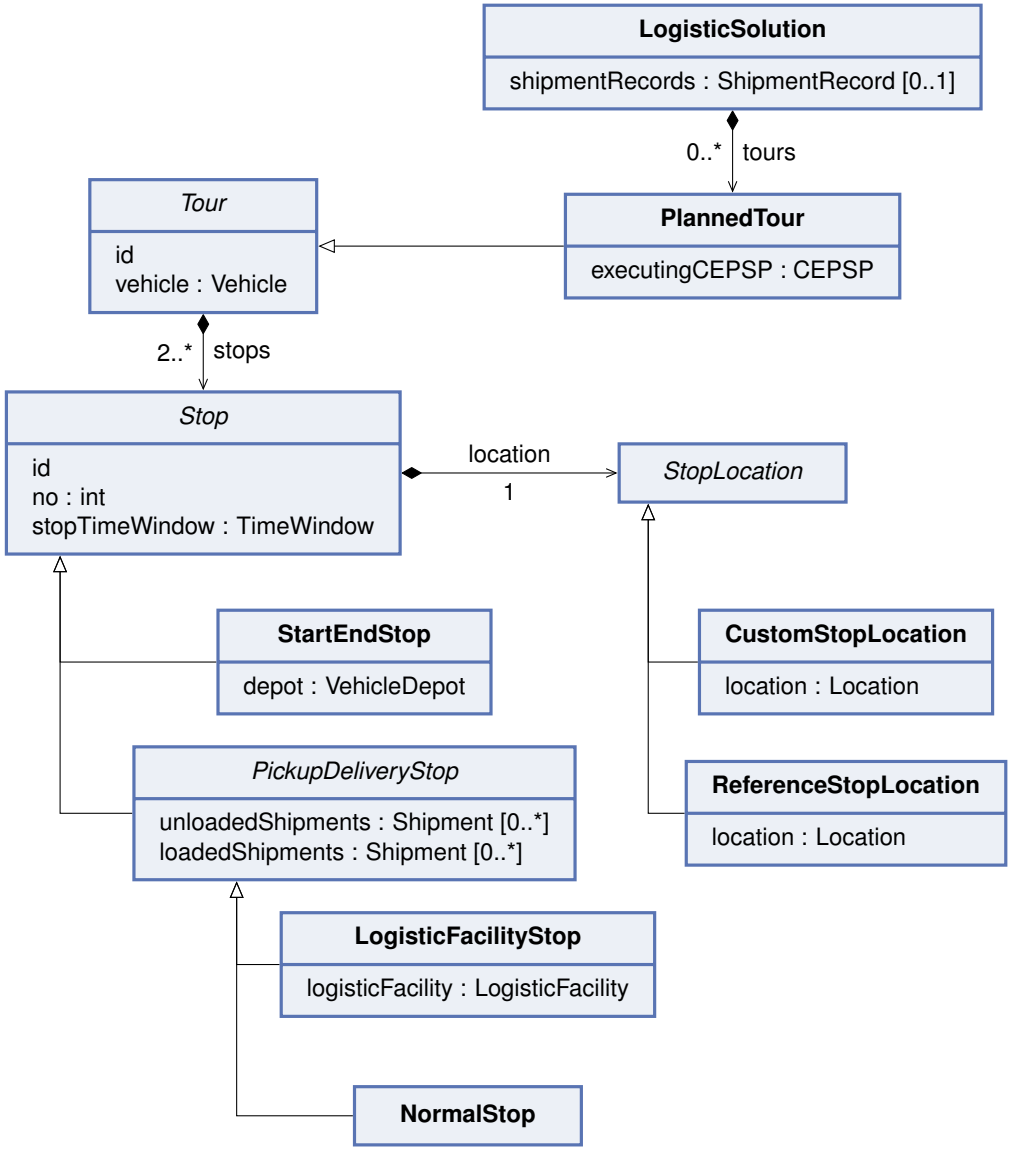


Figure 5.11: Common Metamodel: Logistic Solution View Type - Tours.

The tour structure generally aligns with the tours found in the examined metamodels. However, transformations may need to adjust stops by adding, removing, aggregating, or disaggregating them. For instance, in MATSim-Freight, where only one shipment can be handled per stop, a *PickupDeliveryStop* may need to be disaggregated into multiple stops or vice versa. Another example is the removal or addition of *StartEndStops* when mapping between the logiTopp metamodel and the common metamodel, as the latter lacks an explicit separation between vehicle depots and logistic hubs. Additionally, deriving stop locations poses a challenge for transformations, often requiring a lookup to determine if a stop location is already used anywhere in the logistic network or population.

A shipment record tracks all movements of a shipment, with each entry representing a tour of where the shipment is moved, essentially depicting one hop of the shipment's transport chain. The main metamodel elements for describing shipment records are shown in Figure 5.12. Subsequently, we briefly describe these elements:

- A *ShipmentRecord* has an *id*, a *reference* to the shipment it pertains to, and the *responsibleCEPSP* handling the shipment. Additionally, it encompasses a list of *entries* that chronologically detail the movements of the shipment.
- *ShipmentRecordEntries* have an *id* and an entry number (*no*). They establish links to the *tour* accountable for the shipment record entry, the *executingCEPSP*, and the stops within the tour where the shipment is loaded (*pickUpStop*) and unloaded (*deliveryStop*) from the vehicle. The *timeWindow* specifies the duration during which the shipment is transported, spanning from the start of the pick-up stop's time window to the end of the delivery stop's time window. Furthermore, they contain references to the entity from which the shipment is moved (*fromSpec*) and to which it is moved (*toSpec*).

Note that *ShipmentLegStartEndPoints* constitute a role-based concept defined within the utils package. They denote entities from which a shipment can be transported and to which it can be transported, hence both a *ShipmentConsumerProducer* and a *LogisticFacility* serve as *ShipmentLegStartEndPoints*.

Shipment records also explicitly represent the entry and exit points for shipments. The metamodel elements related to this are shown in Figure 5.13. Subsequently, we briefly describe these elements:

- Within a *ShipmentRecord*, both an *entry* and an *exit* are specified. Each *ShipmentEntry* and *ShipmentExit* includes a *timestamp* to denote when the respective action occurs.
- Three concrete types of *ShipmentEntries* exist. A *LogisticHubEntry* signifies that a shipment enters the study area via a *logisticHub*. *IntermediateEntries* reference the *logisticHub* where the shipment originates in case it has been split, and this is

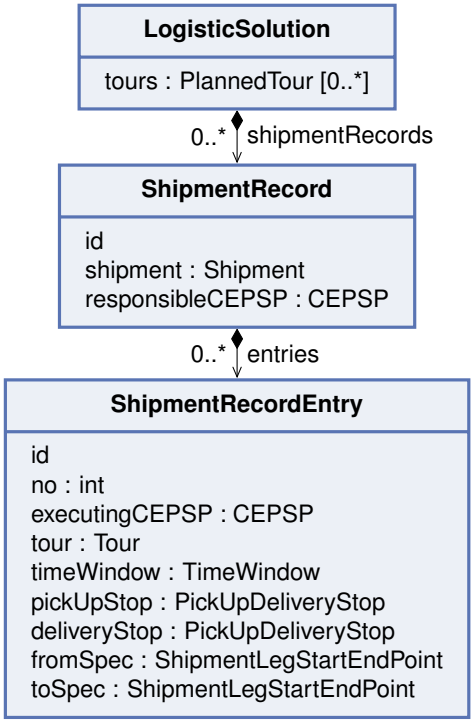


Figure 5.12: Common Metamodel: Logistic Solution View Type - Shipment records.

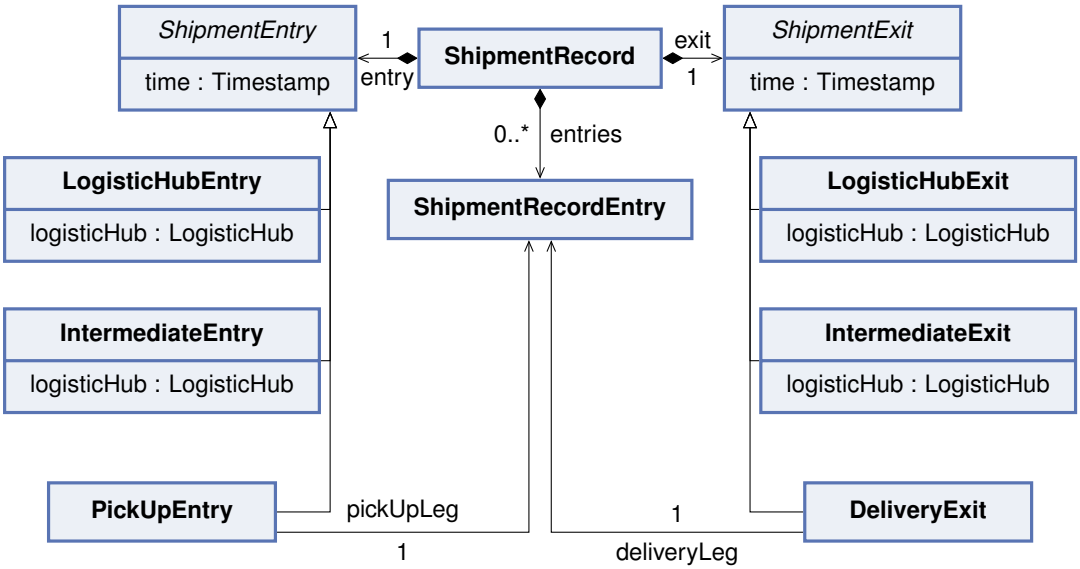


Figure 5.13: Common Metamodel: Logistic Solution View Type - Start and end of shipment records.

not the original entry point. *PickUpEntries* denote instances where the shipment must be picked up initially and include a reference to the corresponding shipment record (*pickUpLeg*)

- Similarly structured to *ShipmentEntries*, *ShipmentExits* encompass three concrete types. A *LogisticHubExit* designates that a shipment exits the study area through a *logisticHub*. *IntermediateExits* refer to the *logisticHub* where the shipment terminates, in scenarios where it has been split and this is not the original exit point. *DeliveryExits* signify situations where the shipment needs to be delivered to its final destination and include a reference to the corresponding shipment record (*deliveryLeg*).

A description of how shipment records can be derived from planned tours is given in Section 5.8.

5.7 Results View Type

In Chapter 4, we initially excluded the results exchange point from active use, as outlined in Section 4.2.3. Furthermore, we acknowledged the necessity for further investigation to determine the required results for implementing feedback loops and their appropriate representation, as discussed in Section 4.2.3. However, both logiTopp and MATSim-Freight feature similar concepts in their result view types, specifically the representation of executed tours. Therefore, despite initial exclusion, we opted to include a representation of executed tours in the common metamodel. We demonstrate how these results can be seamlessly incorporated into the common metamodel with minimal extension of the logistic solution view type.

Similar to the logistic solution view type, the results view type encompasses both tours and shipment records. Figure 5.14 outlines the metamodel elements related to the representation of executed tours.

The *Results* class serves as the root element, containing both *executedTours* and corresponding *shipmentRecords*. An *ExecutedTour* describes an actually executed tour during the simulation, typically referencing the original *plannedTour*. An *ExecutedTour* extends from the abstract class *Tour*, thereby inheriting the same structure. However, the semantics of the included elements undergo slight changes.

Rather than specifying when, where, and what should have occurred during a stop, the elements now describe the actual events that transpired. For instance, an executed stop may deviate from a planned stop in terms of its actual time, influenced by factors such as traffic or delays in previous stops. Additionally, it might not have been possible to deliver a shipment, leading to its return to the logistic hub and, therefore, to an exclusion in the

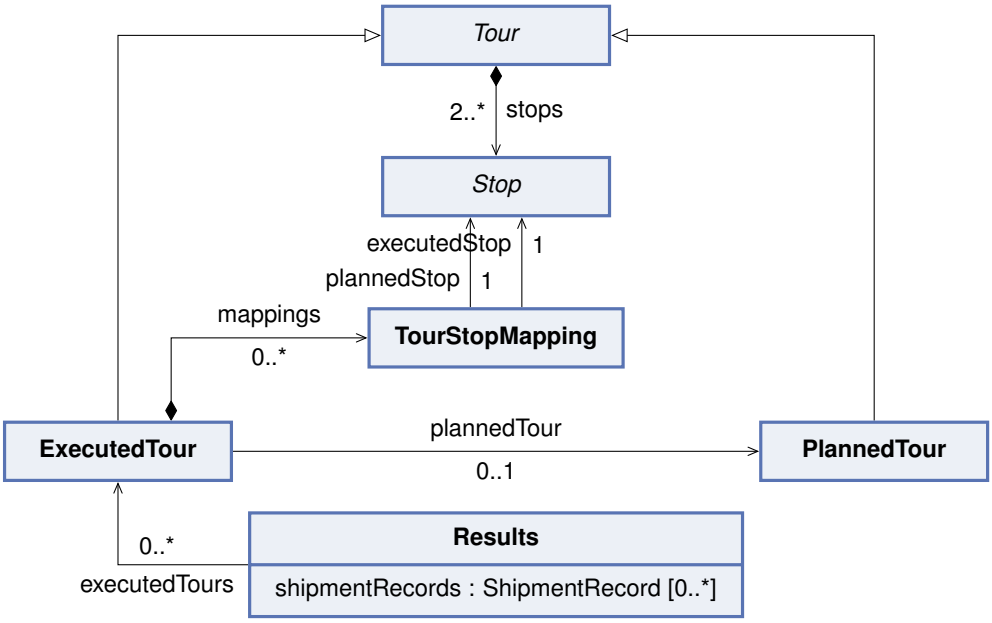


Figure 5.14: Common Metamodel: Results View Type - Executed tours.

unloadedShipments at the current stop and an inclusion in the *unloadedShipments* of the respective *LogisticFacilityStop*.

Furthermore, simulations may introduce modifications such as adding, removing, or altering the order of stops. Consequently, an *ExecutedTour* contains a mapping (*mappings*) that delineates the relationship between a stop of the planned tour (*plannedStop*) and a stop of the *ExecutedTour* (*executedStop*) of preserved stops. Additionally, new tours may be planned during the simulation, in which case the *plannedTour* reference remains unset, and the *mapping* is empty.

The extensions to the shipment records for describing the results of the simulation are shown in Figure 5.15. Instead of utilizing *PickUpEntries* and *DeliveryExits*, the results view type necessitates the use of *ResultPickUpEntries* and *ResultDeliveryExits*. Both *ResultPickUpEntry* and *ResultDeliveryExit* extend the *PickUpEntry* and *DeliveryExit*, respectively, to accommodate failed pickup and delivery attempts. These extensions include references to the *ShipmentRecordEntry* of the successful pickup or delivery, denoted by *pickUpLeg* and *deliveryLeg*, respectively. Moreover, the result extension encompasses a chronological list of references to all previous *FailedPickUpAttempts* or *FailedDeliveryAttempts*, which are themselves subclasses of the *ShipmentRecordEntry*. Given that the location of delivery may change following a failed delivery attempt (e.g., redirecting a shipment to a public service point), the *FailedDeliveryAttempt* records the location where the shipment was initially intended for delivery.

Most simulation-based metamodels do not contain a representation of their results similar to the shipment record structure and only log the executed tours. The logs of

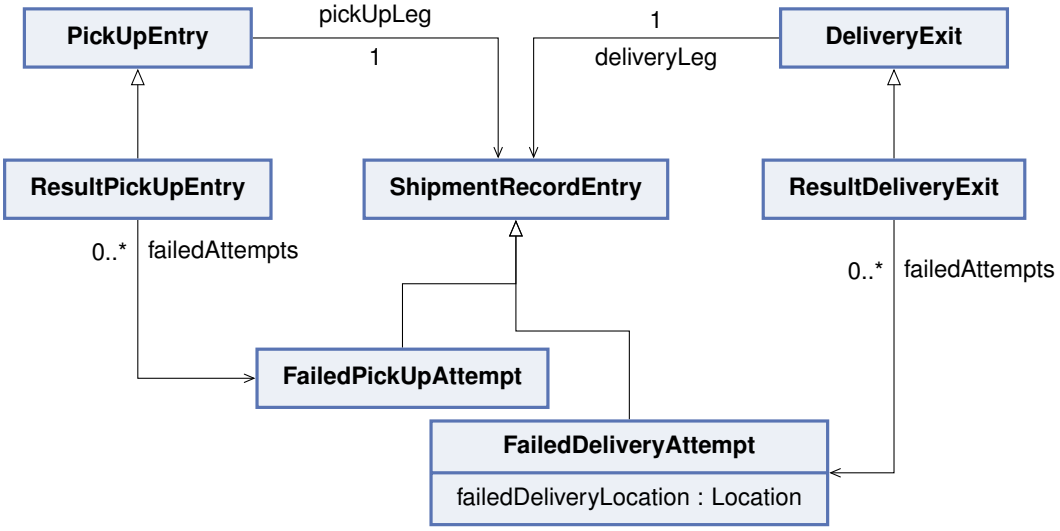


Figure 5.15: Common Metamodel: Results View Type - Shipment records.

executed tours can be used to determine the executed tours of the common metamodel results. However, the tracing between originally planned tours and actually executed tours can be quite challenging if the model allows greater differences between them. If no shipment record-like representation is available, the resulting shipment records can be derived from the executed tours, similar to the derivation of shipment records from planned tours. This is described in Section 5.8.3.

5.8 Transformations for Variation Points

In this section, we discuss the implementation of the transformations aimed at altering variants of the variation points within the common metamodel (also referred to as common transformations). The transformations presented within this section are tailored for application at the logistic solution exchange point for understandability reasons. Adaptations or extensions may be necessary for application at other exchange points. The transformations are presented in an imperative style, consistent with the imperative transformation languages used in our implementation discussed in Chapter 6.

We decided to delve into the implementation details of one variation point, namely the simulation period variation point, to illustrate the complexity of the transformations, the significance of shipment records as a supporting structure, and the general techniques and challenges involved in handling variation points. For the other variation points, we provide a higher-level overview to maintain clarity and focus on key aspects without overwhelming the reader with excessive detail.

5.8.1 Dimension

The dimension variation point encompasses three variants and is implemented using an abstract base class (*Dimension*). Consequently, a transformation simply involves replacing an instance of one subclass representing a variant with an instance of another subclass.

When transitioning to the *Volume-and-Weight-based* variant, either the volume or the weight can be copied into the respective attribute, while the other attribute is set to infinite weight or volume, respectively. Other approaches are possible here. However, this approach, together with the intended semantics of the *Volume-and-Weight-based* variant, seems to retain the constraints of the source model the best. When transitioning from the *Volume-and-Weight-based* variant, either the volume or the weight can be used directly, depending on the selected variant.

Directly transitioning between the *Volume-based* and *Weight-based* variants poses a challenge as there is no fixed relation between volume and weight. Several strategies could be employed here. A straightforward solution involves using a conversion factor, such as the average density of a shipment. An example of such a factor is the dimensional weight proposed by the IATA [26], which is utilized in the context of pricing lightweight but voluminous shipments in air transport.

5.8.2 Network Access

The network access variation point encompasses two variants and is implemented using an abstract base class (NetworkAccess). Consequently, a transformation simply involves replacing an instance of one subclass representing a variant with an instance of the other subclass.

When transitioning to the *Edge-based* variant, any outgoing or incoming edge of the node referenced by the original NodeBasedNetworkAccess could be selected, and the edge position is set to be as close as possible to the original node. More advanced strategies may utilize the coordinates of the location to determine the specific edge and edge position to be used.

Conversely, in the opposite direction, the node of the resulting NodeBasedNetworkAccess is either the origin or destination of the original edge, depending on which node is closer to the position at the edge.

5.8.3 Shipment Records

The shipment records variation point determines whether a representation of shipment records is present in the common metamodel. Thus, additional metamodel elements are introduced to realize this variant. Removing shipment records is straightforward, as they are designed in such a way that no other metamodel elements depend on them.

Creating shipment records for a logistic solution requires no additional input, as they can be derived entirely from the planned tours. For every shipment in the logistic demand, a corresponding shipment record can be created, and its properties can be derived straightforwardly from the respective shipment. To create the shipment record entries, the tours transporting the respective shipment along with the respective pickup and delivery stop must be determined. A shipment record entry must be created from each of these triplets, and most of its attributes can be derived straightforwardly. The fromSpec and the toSpec of a shipment record reference the entities (ShipmentLegStartEndPoints) from and to which the shipment is transported in the tour. If the corresponding tour stop has a reference stop location, this reference can be derived using the respective location's containment relation. Final pickup and delivery can also take place at custom stop locations. In this case, the fromSpec and the toSpec have to be either the consumer or producer of the shipment, depending on whether the shipment is delivered or picked up at the stop. Further, the numbering and order of the shipment record entries must be determined. Which can be achieved by sorting them according to their time window.

5.8.4 Simulation Period

The simulation period variation point allows the common metamodel to either represent a simulation period of a single day, from midnight to midnight, or a whole week (seven days). Consequently, two model transformations are required to map between the variants.

The variation point is primarily realized by the two abstract base classes, Timestamp and TimeWindow, along with their respective subclasses. Furthermore, when transitioning to the *SingleDay* variant, only a subset of the logistic demand and solution can be represented in the target model. We, therefore, require a parameter (*requestedSimulationDay*) to define which day of the simulation should be included in the target model. In the opposite direction, the source model inherently only includes the logistic demand and solution for only one day of the simulation period of the target metamodel. Consequently, the demand and solution for the remaining days must be derived from this single day.

Algorithm 1: SingleToMultiDay

```
1 singleDayDemand ← logisticDemand
2 singleDaySoluton ← logisticSolution
3 removeAllContent(logisticDemand)
4 removeAllContent(logisticSolution)
  // Time windows and opening hours get copied for every simulation day
5 removeAllContent(logisticDemand) removeAllContent(logisticSolution)
6 for simulationDay ← 0 to 6 do
7 |   logisticDemand ← logisticDemand∪
  |    mapTimeElementsToReqSimulationDay(singleDayDemand, simulationDay)
8 |   logisticSolution ← logisticSolution∪
  |    mapTimeElementsToReqSimulationDay(singleDaySoluton, simulationDay)
9 end
```

The pseudocode of a possible transformation from the *SingleDay* to the *MultiDay* variant is outlined in Algorithm 1. In this implementation, the logistic demand and solution for a planned day are duplicated and inserted into the previously emptied logistic demand and solution (lines 6-9). Within each duplicate, the time elements are mapped to the corresponding *MultiDay* element, where the day value is set to the current *simulationDay*. This straightforward approach assumes that shipments are transported entirely within a single simulation day in the source model.

Additionally, two alternative variants are conceivable for this transformation:

- Creating a *MultiDay* model by merging multiple *SingleDay* models. This approach would overcome the previously mentioned restriction, allowing for the merging of shipments and shipment records to reconstruct transport chains spanning multiple days.
- Implementing more sophisticated methods to accommodate demand fluctuations across weekdays. For example, the input model could represent the day with the highest demand, and for other days, only a portion of the shipments corresponding to the demand of each weekday is transferred, with adjustments made to the corresponding logistic solution.

The pseudocode for possible implementation of the reverse transformation is outlined in Algorithm 2. The transformation takes the slice of the logistic solution that falls within the requested simulation day and repairs the logistic solution and demand by modifying and removing elements. Other view types remain unchanged during this process. A crucial aspect of the transformation is defining the simulation day slice accurately. In our proposed implementation, we remove entries from shipment records and corresponding tours if the shipment is transported over midnight or entirely outside the requested simulation day. This process is visually depicted in Figure 5.16. The Figure provides a

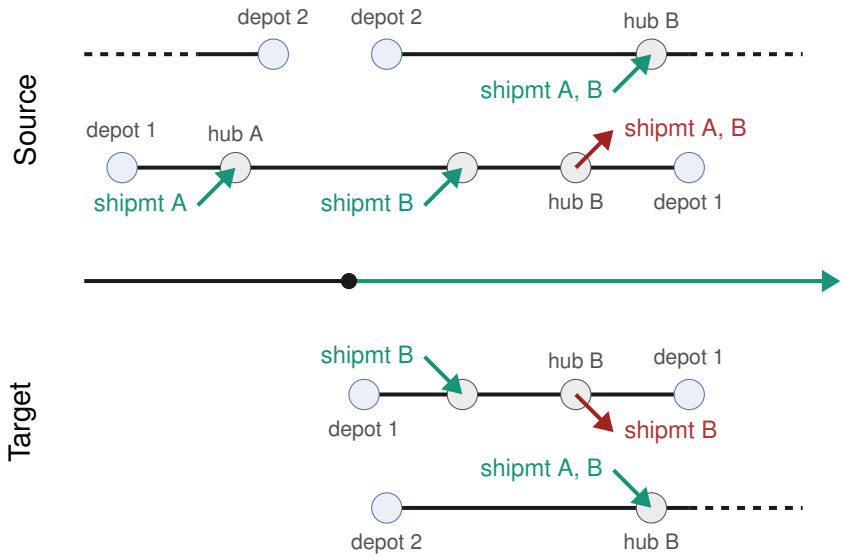


Figure 5.16: Example for the slicing during the transformation from MultiDay to SingleDay variant.

schematic representation of the logistic solution before and after the transformation. The tour ending at depot 2 falls outside the requested simulation day (marked as the green part of the timeline) and is thus removed entirely from the model. Within the tour starting and ending at depot 1, shipment A is removed because it is transported over midnight. However, the same shipment remains in the tour originating from depot 2. Additionally, the start of the tour starting at depot 1 has been adjusted to align with the beginning of the requested simulation day. This heuristic assumption should work well, as last-mile deliveries rarely occur over midnight.

The algorithm begins by updating the shipment records by deleting entries outside the requested simulation day slice while tracking the modified elements. Subsequently, tours and shipments are adjusted to accommodate these changes. Finally, time representations within the model are transformed to the *SingleDay* representation. The algorithm steps are as follows:

- Update shipment records (lines 1-16): Remove entries with time windows outside the simulation day slice. Track deleted entries and adjust entry numbers and id's accordingly. Delete shipment records and corresponding shipments with no remaining entries, as they are irrelevant for the requested simulation day.
- Update planned tours (lines 17-34): Remove the corresponding shipment from the tour for each deleted shipment record entry by adjusting loaded and unloaded shipments at pickup and delivery stops. Remove stops from the tour if they no longer contain any shipments. Eliminate empty tours and move the stop time windows of the first and last stops of the remaining tours within the requested simulation day.

Algorithm 2: MultiToSingleDay(*requestedSimulationDay)*

```
1  deletedEntries ← {}
2  modifiedShipmtRecords ← {}
3  for shipmtRecord in logisticSolution.shipmtRecords do
4      for entry in shipmtRecord.entries do
5          if entry.timeWindow notCompleItyIn requestedSimulatenDay then
6              shipmtRecord.entries ← shipmtRecord.entries \ {entry}
7              deletedEntries ← deletedEntries ∪ {entry}
8              modifiedShipmtRecords ← modifiedShipmtRecords ∪ {shipmtRecord}
9          end
10     end
11     restoreNumbersAndIds(shipmtRecord)
12     if shipmtRecord.entries isEmpty then
13         removeFromModel(shipmtRecord.shipmt)
14         removeFromModel(shipmtRecord)
15     end
16 end
17 for deletedEntry in deletedEntries do
18     shipmt ← deletedEntry.parent.shipmt
19     tour ← deletedEntry.tour
20     pickUpStop ← deletedEntry.pickUpStop
21     deliveryStop ← deletedEntry.deliveryStop
22     pickUpStop.ldShipmts ← pickUpStop.ldShipmts \ {shipmt}
23     deliveryStop.unldShipmts ← deliveryStop.unldShipmts \ {shipmt}
24     if pickUpStop loadedAndUnloadedShipmtsAreEmpty then
25         tour.stops ← tour.stops \ {pickUpStop}
26     end
27     if deliveryStop loadedAndUnloadedShipmtsAreEmpty then
28         tour.stops ← tour.stops \ {deliveryStop}
29     end
30     if tour doesNotContainPickUpDeliveryStops then
31         removeFromModel(tour)
32     end
33     moveStartAndEndStopIntoRequestedDay(tour)
34 end
35 for modifiedShipmtRecord in modifiedShipmtRecords do
36     shortendShipmt ← createShortendShipmt(modifiedShipmtRecord)
37     replaceInModel(modifiedShipmtRecord.shipmt, shortendShipmt)
38 end
39 for shipmt in logisticDemand.shipmts do
40     moveArrivalAtDestinationAndOriginIntoRequestedDay(shipmt)
41 end
42 replaceMultiDayBySingleDayTimes()
```

- Update demand (lines 35-41): Irrelevant shipments for the simulation day have already been removed from the model. However, adjustments are still necessary for the remaining shipments. If a shipment's record has been shortened, the corresponding shipment is replaced within the complete model by a newly created split shipment. This split shipment is essentially a duplicate of the original shipment, referencing the original shipment's id. Further, origin, destination, and arrival times are adjusted according to the shortened shipment record. Finally, all remaining shipments have their arrival times at origin and destination moved to fall within the specified simulation day.
- Update time representing elements (line 42): Convert time specifications in the whole to the *SingleDay* representation by removing the day attribute. Time windows that are partially within the requested simulation day are shortened, while time windows completely outside the requested simulation day are deleted.

This is an example of a common transformation that necessitates a particular variant of the common metamodel. Here, the proposed implementation depends on the presence of shipment records. This highlights how the redundant shipment record structure within the common metamodel facilitates algorithms and transformations.

5.8.5 Transport Chains

The transport chains variation point determines whether the common metamodel contains complex transport chains. A shipment has a complex transport chain if it is transported in more than one tour. Thus, the *Splitted* variant is fulfilled if the following constraint holds for all shipment records: *shipmentRecord.entries.size* ≤ 1. So, variability is only realized by the flexibility of the model and the constraints enforcing a variant, and the variation point is only relevant when exchanging logistic solutions.

To maintain the semantics of a complex transport chain described by the logistic solution in the *Splitted* configuration, we map each hop of a shipment to a separate shipment. In contrast, the invariant of the *Combined* variant is fulfilled by default in the *Splitted* configuration. However, in most application scenarios, it is helpful to recombine shipments that originally represented one shipment again to represent the original transport chains in the model. In the following, we discuss both transformations and possible variants in more detail.

To transfer a common metamodel instance into the *Splitted* configuration, each shipment whose associated shipment record contains more than one entry is split into several SplittedShipments, one for each shipment record entry. Most properties of the newly created shipment can be derived from the original shipment and the corresponding shipment record entry straightforwardly. Further, the consistency of the

model must be maintained. This involves replacing references to the original shipments with the corresponding new shipments in the respective tours. The original shipment should then be removed from the demand while the new shipments are inserted into it. Additionally, a new shipment record must be created for each new shipment.

While most properties of the SplittedShipments can be derived straightforwardly, determining the arrival at origin and arrival at destination time windows presents challenges. Figure 5.17 illustrates various strategies for handling these time windows. The parts in purple shows the original shipment's arrival at the origin (represented by an arrow) and destination time window (depicted as a time interval with bars at the start and endpoint). Further, each hop of the original shipment is shown. The time a shipment takes to be processed at a logistic hub (minimum transshipment time) is denoted curly. Below, different variants are depicted, each showing the resulting new shipments, with associated transports, arrival at origin, and arrival at destination time windows, in different colors. Each variant ensures that the shipment can leave the logistic hub at the earliest after the minimum transshipment time has passed and reach the next destination within a timeframe that allows processing. The difference between the presented variants lies in where, between the arrival and departure of a shipment at a logistic hub within a logistic solution, the time for processing the shipment is placed. Therefore, the variants decide whether potential replanning steps can be more flexible, scheduling the inbound or outbound leg at a logistic hub. Which variant is the most suitable may depend on the actually use case.

To recombine shipments that originally represented one shipment, SplittedShipments contain the id of their original shipment. Consequently, all shipments referencing the same original ID can be merged into one shipment. The merged shipment then replaces references to the split shipments in tours and demand. Additionally, the shipment records have to be merged again.

The back-and-forth transformation between the variants is largely loss-free, with the exception of the arrival time window, which may be shortened depending on the specific strategy.

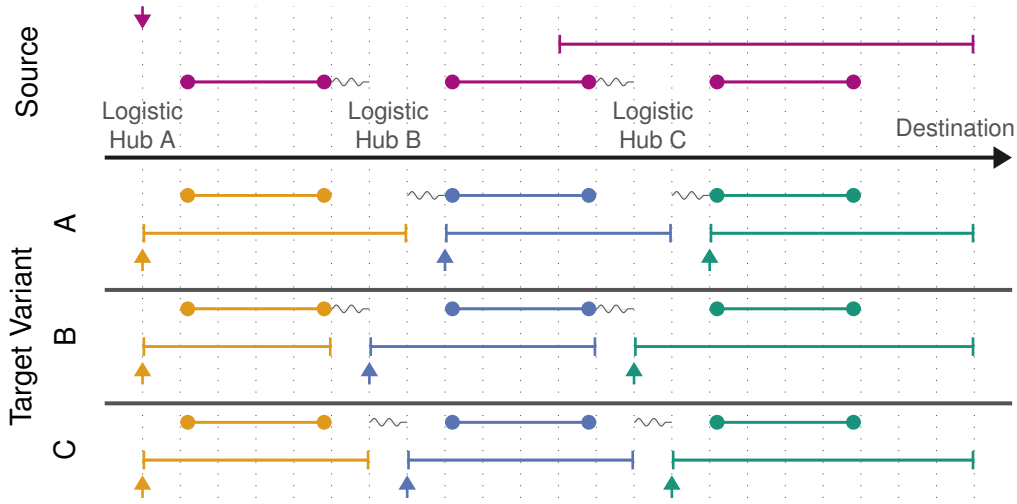


Figure 5.17: Example for splitting a shipment with variants for the determination of the time windows for the resulting shipments.

6 Implementation

Chapters 4 and 5 presented a concept for the coupling of freight transport metamodels along with the therefore utilized common metamodels and some comments on the relationships and transformations between the models studied and created. Additionally, Chapters 2 and 3 provided insights into the freight transport models logiTopp and MATSim-Freight, along with their respective data models in the form of metamodels.

This chapter focuses on the prototypical implementation of the coupling between logiTopp and MATSim-Freight. Specifically, the prototype facilitates the exchange at the logistic solution exchange point (EP2) from logiTopp to MATSim, enabling the simulation and refinement of a logistic solution derived by logiTopp in MATSim-Freight. As discussed in the preceding chapters, minor adjustments could extend this capability to facilitate coupling at the logistic demand exchange point.

This chapter starts with an overview of the implementation in Section 6.1, followed by a brief description of each component implementation, supplemented by some technical details, in Section 6.2.

6.1 Overview

Figure 6.1 provides an overview of the components and process of the implementation, accompanied by the languages utilized for each component. The freight transport models, logiTopp and MATSim-Freight, are represented in green. Both generate or consume a set of files as input and output. The developed components are represented in grey. Metamodels, their instances, and transformations are represented in blue.

Initially, the files generated by logiTopp are parsed by the *LogiToppModelBuilder*, which constructs an instance of the logiTopp metamodel from them. A model-to-model transformation then transforms this instance into an instance of the common metamodel (*LogiTopp2Common*), with corresponding feature configuration. To transition to MATSim-Freight, the feature configuration of the common metamodel instance must be modified. For this purpose, three common transformations are executed in the following order: *BuildShipmentRecords*, *SplitTransportChains*, and *MultiToSingleDay*. Activating the shipment records feature is a prerequisite for the subsequent common transformations.

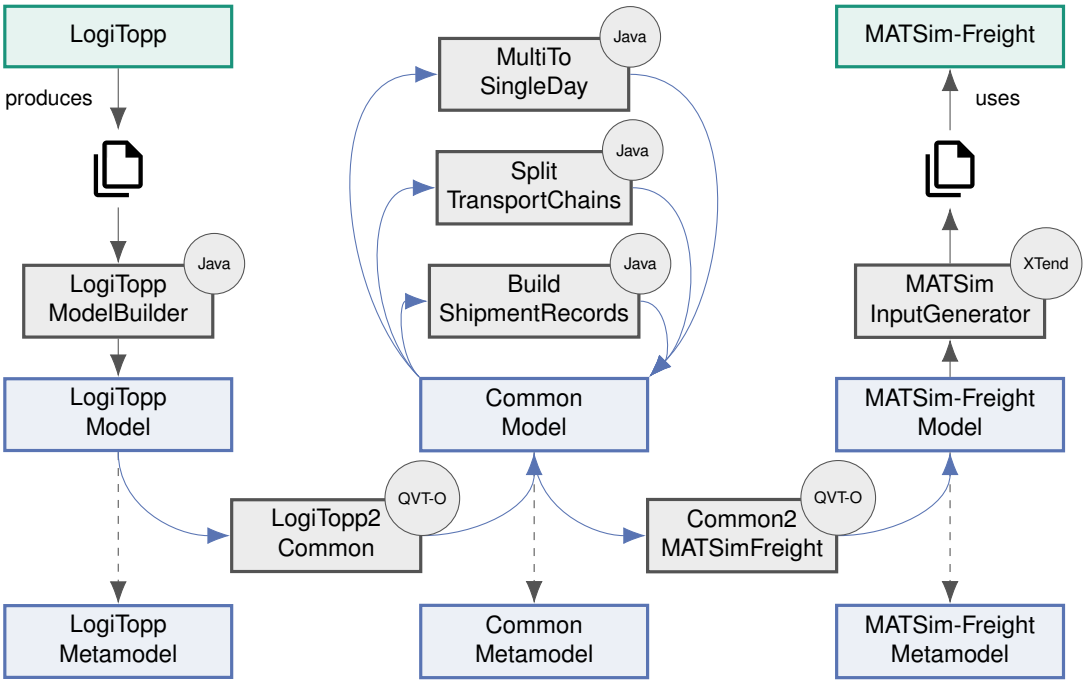


Figure 6.1: Overview of the components and process of the prototypical implementation.

Subsequently, the common metamodel instance is converted to an instance of the MATSim-Freight metamodel (*Common2MATSimFreight*). In the final step, a model-to-text transformation generates the requisite input data for MATSim-Freight.

6.2 Implementation Details

Subsequently, we describe each component of the developed prototype along with some technical considerations and implementation details.

LogiTopp and LogiToppModelBuilder

The *LogiToppModelBuilder* serves as an extraction component within the conceptual architecture (refer to Section 4.1) and is tasked with generating an instance of the logiTopp metamodel at the designated exchange point. However, pinpointing these exchange points in logiTopp poses a challenge due to its dynamic generation of logistic demand and solution during runtime. LogiTopp dynamically creates parcels and plans tours within its simulation loop, triggered by agent reactions to events like parcel arrivals or failed delivery attempts (see also Section 2.3.2). Ultimately, logiTopp generates simulation traces as a product of the simulation.

To capture the initial logistic demand and solution, the logiTopp implementation was slightly modified. The parcel delivery strategy was adjusted to disregard recipient availability, thus ensuring the generated simulation traces accurately reflect the initial planned logistic solution.

Subsequently, the *LogiToppModelBuilder* parses the produced simulation traces and other logiTopp input data, including network, population, and logistic network representations, to create a corresponding instance of the logiTopp metamodel programmatically. Most of this input is provided in CSV format.

Metamodels

The metamodels used within the implementation are the simulation-based metamodels of LogiTopp and MATSim-Freight, described in Chapter 3, and the common metamodel described in Chapter 5. All these metamodels are instances of the Ecore meta-metamodel of the EMF [57].

Model Transformations from and to simulation-based Metamodels

For the implementation, two model-to-model transformations that facilitate the transformation of logiTopp metamodel instances to common metamodel instances (*LogiTopp2Common*) and the transformation of common metamodel instances to instances of the MATSim-Freight metamodel (*Common2MATSimFreight*), each with respective feature configuration, have been developped. Chapter 5 outlined the relationships between these metamodels and how to implement them from a conceptual point of view.

```
1  mapping LOGITOPP::network::Node::node2Node() :
2  COMMON_MM::network::Node {
3    id := self.id;
4    coordinate := self.coord.map point2D2Coordinate();
5  }
6
7  mapping LOGITOPP::network::Point2D::point2D2Coordinate() :
8  COMMON_MM::utils::Coordinate {
9    x := self.x;
10   y := self.y;
11 }
12
13 mapping LOGITOPP::network::Edge::edge2Edge() :
14 COMMON_MM::network::Edge {
15   id := self.id;
```

```
16    from := self.from.resolveone(COMMON_MM::network::Node);
17    to := self.to.resolveone(COMMON_MM::network::Node);
18    length := mapLenght(self.length);
19  }
```

Listing 6.1: Example of a QVT-O transformation mapping network elements.

From a technical perspective, these transformations have been implemented using QVT Operational Mappings (QVT-O) [44]. QVT-O enables the expression of relations and transformations using imperative language constructs and supports various language constructs and query mechanisms commonly found in other high-level programming languages. Additionally, it provides useful mechanisms for model transformations, such as support for object resolution.

One example of a useful feature in QVT-O is the keyword *resolveone*, which is an example of object resolution. It can be applied to an object of the source model and returns the last target object created from the source object through a *mapping*. An example usage of *resolveone* is shown in Listing 6.1, which presents an excerpt of the implemented transformation from the logiTopp metamodel to the common metamodel. For more details on the usage of QVT-O, we refer the reader to the work of Nolte [40].

Common Transformations

The prototype includes three common transformations responsible for adjusting the feature configuration of the common metamodel. The optional shipment records feature is initially added as a precondition for the subsequent transformations (*BuildShipmentRecords*). Subsequently, the variants of the transport chains and simulation period variation points are modified (*SplitTransportChains* and *MultiToSingleDay*). Each of these transformations generates a new instance of the common metamodel, which serves as the source model for the subsequent transformation. Section 5.8 outlined the implementation and algorithms for these transformations from a conceptual perspective.

Although implementing these transformations in a proper model transformation language is feasible, they have been implemented in Java Each of them modifies the loaded model elements in place using the provided API. One significant reason for this decision is the absence of a suitable copy mechanism in QVT-O. Implementing these transformations using QVT-O would result in substantial boilerplate code, as each transformation leaves large portions of the model unchanged. Additionally, many of these common transformations involve complex semantic changes, and Java offers convenient methods for supporting these transformations with additional data structures,

enhancing their clarity, conciseness, and maintainability.

MATSim-Freight and MATSimInputGenerator

In the prototypical scenario, MATSim-Freight is responsible for simulating and potentially optimizing the provided logistic solution. The user can configure this process by specifying MATSim parameters such as the number of iterations and the allowed replanning strategies of carriers and other agents. Typically, the simulation environment includes not only freight processes but also other agents, such as private individuals, allowing for interactions among them. However, the prototype only provides input data relevant to MATSim's freight contribution. These input data include:

- Network file: Describing the network view type of MATSim.
- Carrier vehicle types file: Describing the vehicle types used by carriers.
- Carrier definitions file: Describing carriers with their fleet, the shipments they have to transport, and their plan.

All these files are in XML format and specified either by a document type definition (DTD) or an XML schema definition (XSD). As elements of the network file are referenced by MATSim-Freight-related input files as well as potentially by other input files, the assumed network of all input data must be consistent. This consistency can be ensured by obtaining the network files from the same resource and avoiding changes to the ids during processing.

```
1  def static generateNetwork(Network network) '''
2      <network>
3        <nodes>
4          «FOR node : network.nodes»
5            «generateNode(node)»
6          «ENDFOR»
7        </nodes>
8        <links capperiod="«network.capacityPeriod»"
9               effectivecellsize="«network.effectiveCellSize»"
10              effectivelanewidth="«network.effectiveLaneWidht»">
11         «FOR link : network.links»
12           «generateLink(link)»
13         «ENDFOR»
14       </links>
15     </network>
16   '''
17
18   def static generateNode(Node node) '''
19     <node id="«node.id»" x="«node.coord.x»" y="«node.coord.y»"/>
20   '''
```

```
21
22  def static generateLink(Link link) '''
23    <link id="«generateLinkId(link.id)»" from="«link.from.id»"
24          to="«link.to.id»" length="«link.length»"
25          freespeed="«link.freespeed»" capacity="«link.capacity»"
26          permlanes="«link.nofLanes»" oneway="1"
27          modes="truck, bike, pt"/>
28  '''
```

Listing 6.2: Example of a template-based XTend model-to-text transformation for network elements.

The *MATSimInputGenerator* is responsible for generating the MATSim-Freight-related input files from a provided instance of the MATSim-Freight metamodel, performing a model-to-text transformation. The *MATSimInputGenerator* is implemented in XTend, which offers template expressions for creating templates filled with values derived from the metamodel instance. These templates support expressions for loops and conditions. An example of the generation of MATSim network elements with XTend is shown in Listing 6.2.

7 Evaluation

In Chapters 4 and 5, a concept for coupling multiple freight transport models along with the required common metamodel and its transformations was introduced. Furthermore, Chapter 6 presented a prototypical implementation for the coupling from logiTopp to MATSim-Freight. This chapter is dedicated to evaluating the presented concepts and implementation. We demonstrate and discuss the applicability and benefits of both the concept and implementation by conducting a small case study in Section 7.1. Additionally, we examine the preservation of elementary properties by the model coupling, thus evaluating the correctness of our implementation in Section 7.2.

The two evaluation methods attempt to answer the following questions:

- Can a coupling between two transport models be realized with the presented concept and implementation, and can it leverage the advantages of different transport models?
- Does the coupling preserve elementary properties of the source model in the target model?

In Section 7.3, we identify further evaluation questions that would contribute to the overall assessment.

7.1 Case Study: Rastatt

In this section, we conduct a small case study to illustrate the functional capability of the presented concept and developed implementation. We start by outlining the case study's setup and scenario and then examine the results. We showcase the transfer of an exemplary logiTopp parcel into MATSim and track how MATSim's iterative algorithm modifies the transferred logistic solution over time. Finally, we conclude with a brief discussion of the case study findings.

Setup

In the case study, logiTopp is coupled with MATSim-Freight (in the direction logiTopp to MATSim-Freight) at the logistic solution exchange point, as described in Chapter 6. The

coupled model describes CEP transport in the area of Rastatt (Baden-Württemberg, Germany) and was developed for the research project LogIKTram [8], which evaluates the usage of existing urban rail infrastructure for the CEP sector via cargo trams. The model includes an additional CEPSP, called *ALL*, that operates cargo trams and cargo bikes, which other CEPSPs can utilize to transport parcels with a cargo tram into the city and distribute them from there with cargo bikes and vice versa.

Initially, population synthesis, demand generation, market simulation, and logistic solution generation were performed by logiTopp with a simulation period of one week and 10% of the actual logistic demand. Subsequently, an instance of the logiTopp metamodel was created with the *LogiToppModelBuilder*. Below are key figures regarding the number of model elements of the resulting model:

Rastatt logiTopp Model

- **Network View Type**
  - # nodes: 135570
  - # edges: 293320 (directed)
- **Population View Type**
  - # persons: 33255
  - # businesses: 100
- **Transport Infrastructure View Type**
  - # CEPSPs: 6 + 1
  - # distribution centers: 7 + 7
- **Logistic Demand View Type**
  - # private parcels: 5616
  - # business parcels: 552
- **Logistic Solution View Type**
  - # tours: 527

Thereafter, the model is transformed into several instances of the MATSim-Freight metamodel, one for each day of the simulation period, by applying several model transformations. In the current implementation, this process is semi-automated, meaning the user must start each transformation step separately. However, this limitation is only present in the prototype and can be overcome with minor effort. Additionally, the resulting models of each intermediate step are available as output and can be examined with the EMF editors.

The transformations implemented in QVT-O had unexpectedly long runtimes (around one hour on a standard personal computer). However, we did not investigate this issue further as it falls outside the scope of this work, and runtime is not a critical factor since the presented concept currently does not contain loops or iterative processes.

An additional processing step had to be added to the existing implementation to actually use the resulting model in MATSim. As the input network of the case study provided by logiTopp is not strongly connected, an analysis of the network's connectivity was conducted, and additional edges were added between dead ends and their corresponding components and disconnected components. Once the network of the MATSim-Freight

models is repaired, the actual input files used by MATSim-Freight are created using the *MATSimInputGenerator* for each day of the simulation period.

Apart from the network, carrier, and carrier vehicle files, MATSim-Freight requires an extensive configuration, which includes the definition of scoring functions and allowed strategies for replanning the carrier plans, among other parameters. This configuration heavily depends on the actual use case and research questions for which MATSim-Freight is used. This case study is limited to demonstrating that MATSim-Freight can handle given input data and that the source model's properties are correctly mapped. Therefore, an exemplary configuration based on the example configuration provided by MATSim, presumably resulting from the work of Schröder et al. [53], was used. The configuration is briefly described below:

MATSim-Freight configuration

iterations: 5
plans per carrier: 5

Replanning Strategies

Strategy 1 (*weight 1.0*)

Changes to another plan if that plan is better (probability of change depends on score difference)

Strategy 2 (*weight 0.5*)

Changes start time of each tour randomly by $\pm$ 1.5h (if within bounds)
Then: reroutes tours in space and time

Scoring

Penalization of experienced travel distances and times
Penalization of vehicle usage (dependent on fixed costs of the vehicle type)
Penalization of missed time windows

Furthermore, the case study does not involve the simulation of private persons or any other traffic, meaning the freight agents cannot interact with other agents in the network. To enable this interaction, additional input to MATSim would be necessary, which could potentially be derived from the mobiTopp parts of the logiTopp model used. The coupling and transfer of data related to private persons between MATSim and mobiTopp has been examined by Briem et al. [14].

With the steps mentioned above, the MATSim-Freight simulation could be executed, and the results examined. Subsequently, we describe the representation and transport of an exemplary parcel in the LogiTopp model, the transferred MATSim-Freight model, and the evolved MATSim solution to analyze which properties have changed during this step. Following this, we briefly investigate how the overall planned tours changed

during the MATSim iterations. Finally, we summarize and discuss the findings of the case study.

Transport of an Exemplary Parcel

To demonstrate the transfer of logistic demand and solutions from logiTopp to MATSim-Freight using the developed prototype, we select an exemplary parcel from the case study and analyze its representation and transport in both the source and target models after the transformation. The properties of the selected parcel and its transport in the logiTopp solution are outlined below:

Parcel 2035 in logiTopp

Parcel

- Type: Business Parcel
- Repsonsible CEPSP: GLS
- Arrival at origin: Tue 00:00:00
- Producer: Distribution Center GLS Rheinst. (608730:1)
- Consumer: Business "org. yeti" (25511:1)

Transport

- Tour: 121 (DC: GLS Rheinst. (GLS), Vehicle: 13 (TRUCK))
  - Pickup: StopNo. 0, Tue 14:00:00 - 14:00:00, 608730:1 (GLS Rheinst.)
  - Delivery: StopNo. 1, Tue 14:06:00 - 14:11:00, 608730:1 (HUB_IN GLS Rheinst.)
- Tour: 120 (DC: HUB_IN GLS Rheinst. (ALL), Vehicle: 125 (TRAM))
  - Pickup: StopNo. 0, Wed 08:06:00 - 08:06:00, 608730:1 (HUB_IN GLS Rheinst.)
  - Delivery: StopNo. 1, Wed 08:23:00 - 08:28:00, 71175:2 (HUB Rastatt)
- Tour: 117 (DC: HUB Rastatt (ALL), Vehicle: 31 (BIKE))
  - Pickup: StopNo. 0, Wed 11:51:00 - 11:51:00, 71175:2 (HUB Rastatt)
  - Delivery: StopNo. 3, Wed 12:06:00 - 12:08:00, 25511:1 (Business "org. yeti")

The parcel is transported on three legs (separate tours), using several modes of transport to reach its final destination. As MATSim-Freight is not capable of representing such complex transport chains, the parcel is represented by three shipments in the corresponding MATSim representation of the transferred solution. Note that the shipments are assigned to multiple carriers and only part of the model of the day they are actually transported. The properties of the shipments corresponding to the examined parcel and their transport in the MATSim models are outlined below:

Parcel 2035 in MATSim (initial and *iteration 5*)

Carrier GLS, Tuesday:

Shipments

Id: 2035_0
From: 608730:1, To: 608730:1
Pickup time-window: 00:00:00 - 23:59:59
Delivery time-window: 00:00:00 - 14:11:00

Transport

Tour: 121 (Vehicle: 13 (TRUCK))
Pickup: 14:00:00 - 14:00:00 | *12:59:50 - 12:59:50*
Delivery: 14:10:30 - 14:10:48 | *13:04:20 - 13:04:38*

Carrier ALL, Wednesday:

Shipments

Id: 2035_1
From: 608730:1, To: 71175:2
Pickup time-window: 00:00:00 - 23:59:00
Delivery time-window: 00:00:00 - 08:28:00

Id: 2035_2
From: 71175:2, To: 25511:1
Pickup time-window: 08:28:00 - 23:59:59
Delivery time-window: 00:00:00 - 23:59:59

Transport

Tour: 120 (Vehicle: 125 (TRAM))
Pickup: 08:06:00 - 08:06:00 | *07:30:00 - 07:30:00*
Delivery: 08:27:30 - 08:27:48 | *07:34:30 - 07:34:48*

Tour: 117 (Vehicle: 31 (BIKE))
Pickup: 11:51:00 - 11:51:00 | *09:49:20 - 09:49:20*
Delivery: 12:06:00 - 12:06:00 | *09:53:21 - 09:55:21*

This example highlights that the delivery and pickup time windows in the resulting MATSim model are structured in such a manner that a shipment must always arrive at an intermediate location before it can be picked up for further transportation. It is conspicuous that the delivery time window of shipment *2035_0* already ends at *14:11:00*. This is a result of the selected strategy of splitting the transport chains (in this case, variant B of Figure 5.17) and the sequence of applied common transformations (first *SplitTransportChains*, then *MultiToSingleDay*).

Additionally, the times of the tour stops are provided. In some instances, the stop duration is shorter than the corresponding stop in logiTopp. This discrepancy arises because a stop in MATSim can only accommodate one shipment. Consequently, when

multiple parcels are handled at a stop in logiTopp, the corresponding stop time window is divided.

Furthermore, the times of the tours optimized by MATSim after iteration 5 are presented in italics. The MATSim algorithm adjusted the times of the stops while adhering to the constraints of the respective time windows. This behavior is attributed, among other things, to the configured scoring function, which penalizes non-compliance with time windows.

Development of Planned Tours over Multiple Iterations

Figure 7.1 illustrates the evolution of the carrier plans between the initial plan and the active plan after five iterations by plotting the current active tours of some carriers against the time axis. A comparison of both plots reveals the impact of the implemented replanning strategies. Specifically, the tours of certain carriers undergo random shifts along the time axis, effectively reducing the peaks of concurrently active tours.

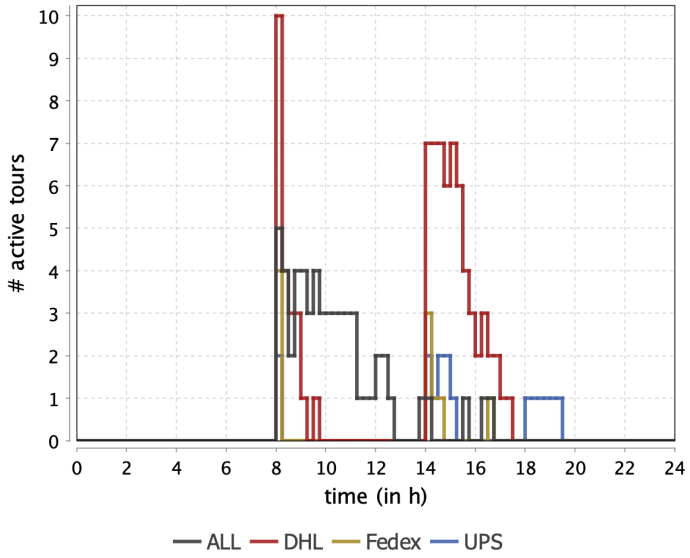
Discussion

The case study confirmed the functionality of the developed prototype within the tested scenario, requiring only minor adjustments to execute, optimize, and analyze the original logiTopp solution in MATSim-Freight. Most adjustments were necessitated by varying requirements for network graph connectivity between the coupled models. However, it remains unclear whether the transformations are responsible for ensuring network graph connectivity and consistency or if this should be a precondition met by the source model.

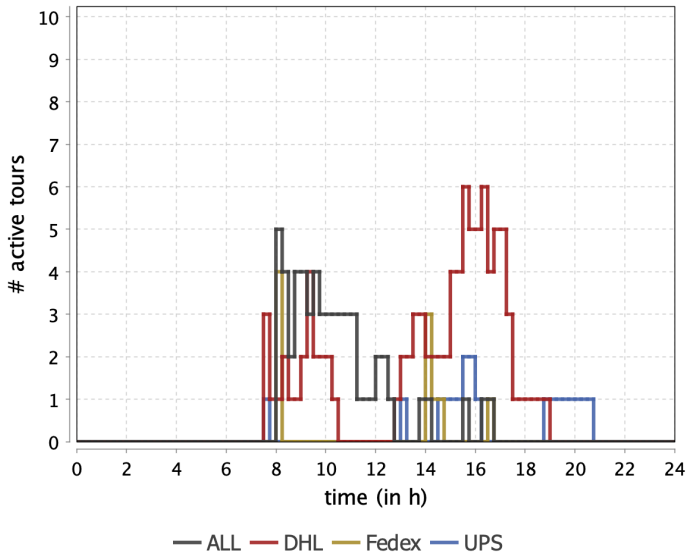
Despite being semi-automated, the process represents a significant improvement over manually creating a corresponding MATSim model regarding user effort, especially considering the potential need for repetitive execution in research processes. An easily achievable enhancement to the prototype would be full automation of the coupling process.

Furthermore, the example parcel demonstrated that the fundamental properties of the original logiTopp solution are accurately transferred to the target MATSim model. This aspect is further explored in Section 7.2.

However, the case study lacked a concrete research scenario, thus failing to illustrate how coupling both models can benefit an actual research process. Nevertheless, it's evident that both models possess distinct strengths and features, suggesting that their coupling can indeed contribute to addressing real-world research questions.



(a) Initial plan.



(b) Selected plan after iteration 5.

Figure 7.1: Change of active tours by time of selected carriers between initial plan and iteration 5 on Wednesday.

7.2 Relational Constraints Between LogiTopp and MATSimFreight

This section aims to evaluate the correctness of the presented concept and especially its implementation. To achieve this, a set of constraints describing relations that must hold between a logiTopp source model and the corresponding MATSim target model created through coupling at the logistic solution exchange point, as described in the previous chapters, is introduced. These constraints should not only benefit the correctness evaluation but also help to gain a deeper understanding of how elements between logiTopp and MATSim are mapped and how the change in the feature configuration influences their relations.

Due to the high effort, the amount of constraints is restricted but should cover the most essential properties of both models. The selection and definition of constraints were approached by systematically examining each viewpoint and identifying which concepts of the source model are contained within or influence the target model. Thereby, the constraints focus on key elements and concepts, paying particular attention to attributes affected by the change of the feature configuration, as these areas often have high complexity and pose a high potential for errors.

Evaluated Constraints

We will now begin by delineating the set of constraints employed for the evaluation, accompanied by the rationale behind their selection and design. In the formal presentation of the constraints, we simplified the treatment of time conversions and the comparison of floating-point numbers. It's important to note that we assumed the models under evaluation are valid instances of their respective metamodels, particularly concerning static semantics. Therefore, we did not explicitly verify them and designed the constraints under the assumption of valid static semantics.

Regarding the network viewpoint, the elementary concept is the directed network graph with its edges and nodes. We introduced the constraints 1 and 2 to evaluate their mapping. After the network view type of the common metamodel is defined in a lean fashion, no further concepts are transferred and thus have not to be checked. Location used in the context of other entities will be checked during their verification.

Constraint 1 (*Network Nodes*)

For every *RoadNetwork Node* $n_{logiTopp}$ in logiTopp, there exists a corresponding *Network Node* n_{MATSim} in MATSim, and vice versa, such that:

- $n_{logiTopp}.id = n_{MATSim}.id$ *(correspondence)*
- $n_{logiTopp}.x = n_{MATSim}.x$

- $n_{logiTopp}.y = n_{MATSim}.y$
- $0 = n_{MATSim}.z$

Constraint 2 (*Network Edges*)

For every *RoadNetwork Edge* $e_{logiTopp}$ in logiTopp, there exists a corresponding *Network Link* l_{MATSim} in MATSim, and vice versa, such that:

- $e_{logiTopp}.id = l_{MATSim}.id$ *(correspondence)*
- $e_{logiTopp}.from.id = l_{MATSim}.from.id$
- $e_{logiTopp}.to.id = l_{MATSim}.to.id$
- $e_{logiTopp}.length = l_{MATSim}.length$

In the implementation, no elements are created from the population viewpoint of MATSim; thus, their existence must not be checked. However, the location of businesses and persons is reflected in the origins and destination of shipments. Thus, we added the constraints 3, 4, and 5 that validate the origins and destinations of shipments if they should reflect the respective location of the population.

Constraint 3 (*Population Location: Person*)

For every *PrivateParcel* $p_{logiTopp}$ received by a *Person* in logiTopp, the consumer's *Location* in the *PrivateParcel* is equal to the destination of the corresponding MATSim *CarrierShipment* s_{MATSim} describing the final delivery:

- Prefix of $s_{MATSim}.id$ start with $p_{logiTopp}.id$ (correspondence)
- s_{MATSim} must describe the final hop of $p_{logiTopp}$'s transport chain
- $p_{logiTopp}.consumer.location.roadAccessEdge.id = s_{MATSim}.to.id$ (location equality)
- Only if:
  - $p_{logiTopp}.destinationType = HOME$
  - $p_{logiTopp}$'s delivery leg is entirely within the *requestedSimulationDay*

Constraint 4 (*Population Location: Business I*)

For every *BusinessParcel* $p_{logiTopp}$ received by a *Business* in logiTopp, the consumer's *Location* in *BusinessParcel* is equal to the destination of the corresponding MATSim *CarrierShipment* s_{MATSim} describing the final delivery:

- Prefix of $s_{MATSim}.id$ start with $p_{logiTopp}.id$ (correspondence)
- s_{MATSim} must describe the final hop of $p_{logiTopp}$'s transport chain
- $p_{logiTopp}.consumer.location.roadAccessEdge.id = s_{MATSim}.to.id$ (location equality)
- $p_{logiTopp}$'s delivery leg is entirely within the *requestedSimulationDay*

Constraint 5 (*Population Location: Business II*)

For every *BusinessParcel* $p_{logiTopp}$ sent by a *Business* in logiTopp, the producers *Location* in *BusinessParcel* is equal to the origin of the corresponding MATSim *CarrierShipment* s_{MATSim} describing the initial pickup:

- Prefix of $s_{MATSim}.id$ start with $p_{logiTopp}.id$ (correspondence)
- Initial pickup: s_{MATSim} must describe the first hop of $p_{logiTopp}$'s transport chain
- $p_{logiTopp}.producer.location.roadAccessEdge.id = s_{MATSim}.from.id$ (location equality)
- Only if: $p_{logiTopp}$'s pickup leg is entirely within the *requestedSimulationDay*

Note that these three constraints are influenced by the transformations required to change the feature configurations as shipments get split, and potentially, not all parts of a transport chain are included in the destination model. As shipments are split, the correspondence between parcels of the source model and shipments of the destination model is a $1..n$ relation. Thus, the split shipments have an id composed of the original id and a suffix describing the split shipment's number. Then, for the test of a shipment's initial origin or final destination, only the first or the last corresponding shipment is required. Further, the pickup or delivery may not fall into the *requestedSimulationDay* so that the corresponding shipment has been removed.

Regarding the transport infrastructure viewpoint, the only elements that are contained in both metamodels are the CEPSPs or carriers and the delivery vehicles or carrier vehicles. Other concepts, such as distribution centers, vehicle depots, and allowed transport chains, can only be implicitly represented in MATSim through the logistic demand and logistic solution view type. This is why we describe the constraints that check parts of these concepts hereafter. Constraint 6 serves as a basic check for the accurate mapping of CEPSPs to carriers. Constraint 7 tests the correct mapping of delivery vehicles and thereby checks some other properties of the transport network. Despite elementary properties such as correspondence, capacity, vehicle type, and operation hours, the mapping to the correct CEPSP and the location of the vehicle is also tested, implicitly testing the mapping of logiTopp elements such as fleets and distribution centers.

Constraint 6 (*CEPSPs*)

For every *CEPServiceProvider* $c_{logiTopp}$ in the logiTopp transport network, a corresponding *Carrier* c_{MATSim} in MATSim exists and vice versa:

- $c_{logiTopp}.id = c_{MATSim}.id$ (correspondence)

Constraint 7 (*Delivery Vehicles*)

For every *DeliveryVehicle* $v_{logiTopp}$ in the logiTopp transport network, a corresponding *CarrierVehicle* v_{MATSim} in MATSim exists and vice versa, such that:

- $v_{logiTopp}.id = v_{MATSim}.id$ (correspondence)
- $v_{logiTopp}.capacity = v_{MATSim}.type.capacity.other$
- $v_{logiTopp}.vehicleType$ correspondsTo $v_{MATSim}.type.networkMode$
- $v_{logiTopp}.fleet.distributionCenter.location.roadAccessEdge.id = v_{MATSim}.location.id$
- $v_{logiTopp}.fleet.distributionCenter.CEPServiceProvider.id = v_{MATSim}.carrierCapabilities.carrier.id$

To efficiently evaluate and express the constraints related to the logistic demand and logistic solution viewpoints, we created a helper data structure similar to the shipment records in the common metamodel, which is described subsequently.

Data Structure(*ParcelRecord and ParcelRecordEntry*)

For every *Parcel* in logiTopp a *ParcelRecord* is created with the following properties:

- *parcel* : *Parcel*
- *entries* : list of corresponding *ParcelRecordEntries*

For every hop of a *Parcels* transport chain in logiTopp a *ParcelRecordEntry* is added to the corresponding *ParcelRecord* with the following properties:

- *record* : *ParcelRecord*
- *no* : int, number of the entry
- *previous* : *ParcelRecordEntry*, previous entry if present
- *next* : *ParcelRecordEntry*, next entry if present
- *tour* : *PlannedDeliveryTour*
- *pickUp* : *ParcelActivity*
- *delivery* : *ParcelActity*
- *start* : *Time*, start time of the pick-up activity
- *end* : *Time*, end time of the delivery activity
- *inSlice* : boolean, true if *start* and *end* are within the *requestedSimulationDay*

The *ParcelRecords* are derived from the logistic solution of the logiTopp source model.

Especially through the transformations required for changing the feature configuration, the information of the transport network, logistic demand, and logistic solution viewpoints is diffused around multiple model elements of the respective viewpoints. An example of this is the transport times assigned by a logistic solution in the planned tours,

which are distributed through this process to the pickup and delivery time windows of the respective split shipments. For this reason, the constraints evaluating properties of target model elements test properties of multiple viewpoints at once. Therefore, we created constraints on the most essential elements of these viewpoints, namely shipments, tours, and tour elements.

The constraints 8 and 9 refer to *CarrierShipments* and *CarrierServices* in the MATSim models. Regarding the *CarrierShipments*, we evaluate a correct mapping of the shipment size and the origin and destination locations of each hop. Further, the *CarrierShipments* must be assigned to the *Carrier* that is responsible for the transport of the respective hop. The constraints regarding the pickup and delivery time windows could be strengthened to assert the concrete variant chosen for the determination of these time windows during the split of transport chains. The presented formulation just requires that a shipment arrive at an intermediate point before the subsequent shipment can be picked up at that point.

Constraint 8 (*Shipments*)

For every *ParcelRecordEntry* $e_{logiTopp}$ with $e_{logiTopp}.inSlice$ derived from the logiTopp model a corresponding *CarrierShipment* s_{MATSim} exists in MATSim and vice versa, such that:

- $e_{logiTopp}.record.parcel.id + "\_" + e_{logiTopp}.no = s_{MATSim}.id$ (correspondence)
- $e_{logiTopp}.record.parcel.shipmentSize$ correspondsTo $s_{MATSim}.size$
- $e_{logiTopp}.pickUp.stopLocation.roadAccessEdge.id = s_{MATSim}.from.id$
- $e_{logiTopp}.delivery.stopLocation.roadAccessEdge.id = s_{MATSim}.to.id$
- $e_{logiTopp}.tour.depotStorage.distributionCenter.CEPServiceProvider.id = s_{MATSim}.carrier.id$
- (pickupTimeWindow)
  - if $e_{logiTopp}.previous$ present:
    $e_{logiTopp}.previous.end \leq s_{MATSim}.pickupTimeWindow.start$
  - else:
    $e_{logiTopp}.record.parcel.plannedArrivalDate = s_{MATSim}.pickupTimeWindow.start$
- (deliveryTimeWindow)
  - if $e_{logiTopp}.next$ present:
    $e_{logiTopp}.next.start \geq s_{MATSim}.deliveryTimeWindow.end$
  - else:
    $s_{MATSim}.deliveryTimeWindow.end > s_{MATSim}.pickupTimeWindow.start$

As there is no mapping from any *Parcel* to a *CarrierService*, Constraint 9 ensures that no *CarrierServices* exists in the target model.

Constraint 9 (*Services*)

No *CarrierServices* exists in MATSim.

Constraint 10 is dedicated to the planned tours of a logistic solution. We require all and only the tours that transport a *Parcel* entirely within the *requestedSimulationDay* to exist in the MATSim model. Further, the constraint checks for the correct assignment of the vehicles to the tour. We do not require equivalence of the tour start times, as tours can be shortened by transforming the simulation period feature.

Constraint 10 (*Tours*)

For every *PlannedDeliveryTour* $t_{logiTopp}$ in the logiTopp model that is referenced by any *ParcelRecordEntry* $e_{logiTopp}$ with $e_{logiTopp}.inSlice = true$ a corresponding *ScheduledTour* t_{MATSim} in MATSim exists and vice versa, such that:

- $t_{logiTopp}.id = t_{MATSim}.tour.id$ (correspondence)
- $t_{logiTopp}.stops[1].vehicle.id = t_{MATSim}.vehicle.id$

Finally, constraints 11 and 12 test the correct mapping of stops for pickup and delivery of parcels. The constraints ensure that every relevant but no further stops exist in MATSim. A stop (*ParcelActivity*) of the logiTopp is relevant if it handles a shipment that is entirely transported within the *requestedSimulationDay* during the respective tour. Further, the constraints enforce that the stops are within the correct tour and that the times of the target model stops are within the boundary of the corresponding source model stop times without enforcing the detailed implementation of disaggregating *ParcelActivities* into MATSim *Deliveries* and *Pickups*.

Constraint 11 (*Tour Elements: Pickup Stop*)

For every *ParcelRecordEntry* $e_{logiTopp}$ in the logiTopp model with $e_{logiTopp}.inSlice = true$ a corresponding *Pickup* p_{MATSim} in MATSim exists and vice versa, such that:

- $e_{logiTopp}.record.parcel.id + "\_" + e_{logiTopp}.no = p_{MATSim}.shipment.id$ (correspondence)
- $e_{logiTopp}.tour.id = p_{MATSim}.tour.id$
- $e_{logiTopp}.pickUp.plannedTime \leq p_{MATSim}.expectedArrivalTime \leq (e_{logiTopp}.pickUp.plannedTime + e_{logiTopp}.pickUp.deliveryDuration)$
- $subsequentLeg(p_{MATSim}).expectedDepartureTime \leq (e_{logiTopp}.pickUp.plannedTime + e_{logiTopp}.pickUp.deliveryDuration)$

Constraint 12 (*Tour Elements: Delivery Stop*)

For every *ParcelRecordEntry* $e_{logiTopp}$ in the logiTopp model with $e_{logiTopp}.inSlice = true$ a corresponding *Delivery* d_{MATSim} in MATSim exists and vice versa, such that:

- $e_{logiTopp}.record.parcel.id + "\_" + e_{logiTopp}.no = d_{MATSim}.shipment.id$ (corre-

spondence)

- $e_{logiTopp}.tour.id = d_{MATSim}.tour.id$
- $e_{logiTopp}.delivery.plannedTime \leq d_{MATSim}.expectedArrivalTime \leq (e_{logiTopp}.delivery.plannedTime + e_{logiTopp}.delivery.deliveryDuration)$
- $subsequentLeg(d_{MATSim}).expectedDepartureTime \leq (e_{logiTopp}.delivery.plannedTime + e_{logiTopp}.delivery.deliveryDuration)$

The constraints have been implemented in Java by querying the source and target metamodel instances and asserting the constraints.

Results and Discussion

The constraints have been evaluated on the models of the Rastatt case study (see Section 7.1) on each *requestedSimulationDay*.

Through this evaluation process, we identified and addressed several minor errors in the prototype implementation. Furthermore, we uncovered two types of errors in the input data provided by logiTopp and, consequently, in the logiTopp implementation itself:

- The tour planning algorithm in logiTopp erroneously clusters parcels with the same destination solely based on the consumer, disregarding the destination type of the parcel. This oversight leads to incorrect routing of parcels with different destination types but the same consumer, such as delivering both parcels to the consumer's household instead of directing one to a packstation.
- The logiTopp implementation neglects the delivery duration of parcel activities, resulting in some parcels being transported immediately after the start of the arrival parcel activity, even if the delivery duration has not elapsed entirely.

After filtering out these two errors, all described constraints were satisfied across the evaluated models.

The explicit formulation of the constraints enhanced the comprehension of the relationships between the two coupled models and elucidated which properties are preserved during the coupling process. However, it is essential to note that while this Unit-Test-like approach provides valuable insights, it cannot guarantee or conclusively prove the correctness of the implementation. Nonetheless, the evaluation demonstrated the correctness of the applied transformations to a significant extent.

7.3 Further Evaluation Questions

In this section, we briefly elaborate on further research questions that would benefit the overall evaluation of the concept presented in this work and could not have been answered with the evaluation methods employed.

A crucial investigation relates to the assessment of the quality and suitability of the common metamodel introduced in this work. Conducting an extensive literature review or engaging in expert interviews could provide insights into whether the common metamodel accurately and adequately captures the CEP or freight transport domain. Furthermore, evaluating the metamodel's understandability and level of abstraction is crucial to ensuring its practical utility. Integrating additional freight transport models into the proposed approach could yield valuable insights into these aspects, thereby enhancing the understanding of the metamodel's effectiveness in fulfilling its pragmatic property.

A comprehensive quality assessment of the employed metamodels and transformations, aligned with established EMF best practices and coding standards, is also missing. While we have demonstrated the correctness of the transformations to some extent, evaluating aspects such as maintainability and adherence to industry standards is essential for ensuring the long-term viability of the prototype.

Another area requiring further exploration is the actual applicability of the developed concept. Does the approach genuinely enhance actual research processes and deliver the expected added value? It is essential to assess whether the provided exchange points and the thereby exchanged information are adequate and beneficial for the use cases. Furthermore, comparing the proposed approach to other existing methodologies and the current state-of-the-art research processes would provide valuable insights into its effectiveness and potential advantages.

8 Conclusion and Future Work

In this final chapter, we conclude the thesis and give an overview of possibilities for future work.

8.1 Conclusion

This thesis introduced a concept for coupling multiple freight transport models to leverage their diverse features within an integrated workflow. Central to this concept is the development of a common metamodel for the freight transport domain, which serves as a central component for data exchange through model transformations among the coupled freight transport models.

This work started with a comprehensive analysis of existing freight transport models, focusing on logiTopp and MATSim-Freight, particularly investigating their processes, concepts, and underlying data models. Subsequently, we presented the developed concept and common metamodel, addressing two primary challenges. Firstly, we tackled the heterogeneity of processes and incorporated concepts within freight transport models by defining criteria for suitable exchange points. This facilitated the derivation of a standardized model process with predefined points and associated content for data exchange. Secondly, we addressed the challenge of handling the variability in the data models and employed concepts among coupled freight transport models by introducing variable components into the common metamodel. Leveraging a set of transformations on the common metamodel, we ensured that a common metamodel instance was adapted to the specific needs of both the source and target models, thereby preserving elementary properties through the coupling process.

A prototype for coupling logiTopp and MATSim-Freight was developed to evaluate the proposed concept. A case study demonstrated the prototype's functionality, serving as a proof of concept. Additionally, a set of constraints describing the preservation of elementary properties through coupling both models was presented. While the evaluation provided insights into the prototype's correctness to some extent, further investigation is warranted to assess other aspects comprehensively.

In essence, the primary outcome of this work is a concept that aims to couple multiple

freight transport models instead of just a pair of two models and a common metamodel that facilitates both coupling and a shared understanding of the domain.

8.2 Future Work

Within the thesis, ideas and research questions for future work have been pointed out, which are summarized in this section.

Several extensions to the concept are discussed in this thesis. First, the merging of common metamodels at exchange points. Second, the integration of iterative processes and feedback loops. Both extensions would enhance the flexibility and support more use cases within the approach.

Additionally, integrating further freight transport models into the common model and prototype would provide insights into assessing the quality and applicability of the common metamodel. For each integrated freight transport model, the common metamodel must be further developed, clarifying which concepts of the new model are already part of the common model and which concepts match or mismatch, potentially leading to additional variability.

Furthermore, the proposed approach's application to agent-based transport models in general is of interest and can broaden the concept's scope. Therefore, it must be investigated whether these models also have common concepts and viewpoints to enable coupling through a common metamodel, and further challenges arise in this area of application. Finally, the research and development of alternative approaches to the coupling of freight transport models would enrich the field, clarify the advantages and disadvantages of the developed approach, and offer further perspectives on model coupling.

Bibliography

[1] graphhopper/jsprit: jsprit is a java based, open source toolkit for solving rich vehicle routing problems, . URL https://github.com/graphhopper/jsprit.

[2] matsim-org/matsim-libs: Multi-Agent Transport Simulation, . URL https://github.com/matsim-org/matsim-libs.

[3] M. Adnan, F. C. Pereira, C. L. Azevedo, K. Basak, M. Lovric, S. Raveau, Y. Zhu, J. Ferreira, C. P. Zegras, and M. E. Ben-Akiva. SimMobility: A Multi-scale Integrated Agent-Based Simulation Platform. volume 2, 2016. URL https://api.semanticscholar.org/CorpusID:111537723.

[4] Aimsun. *Aimsun Next 24 User's Manual*. Barcelona, Spain, aimsun next 24.0.0 edition, 2024. URL https://docs.aimsun.com/next/24.0.0.

[5] A. Alho, B. Bhavathrathan, M. Stinson, R. Gopalakrishnan, D.-T. Le, and M. Ben-Akiva. A multi-scale agent-based modelling framework for urban freight distribution. *Transportation Research Procedia*, 27:188–196, 2017. ISSN 23521465. doi: 10.1016/j.trpro.2017.12.138. URL https://linkinghub.elsevier.com/retrieve/pii/S2352146517310359.

[6] S. Apel, D. Batory, C. Kästner, and G. Saake. *Feature-Oriented Software Product Lines: Concepts and Implementation*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-37520-0 978-3-642-37521-7. doi: 10.1007/978-3-642-37521-7. URL https://link.springer.com/10.1007/978-3-642-37521-7.

[7] K. W. Axhausen and ETH Zürich. *The Multi-Agent Transport Simulation MATSim*. Ubiquity Press, Aug. 2016. ISBN 978-1-909188-75-4. doi: 10.5334/baw. URL http://www.ubiquitypress.com/site/books/10.5334/baw/.

[8] L. Barthelmes, E. Görgülü, J. Kübler, M. Kagerbauer, and P. Vortisch. Modellbasierte Ermittlung von verkehrlichen Potentialen eines stadtbahnbasierten Gütertransports im Projekt LogIKTram in Karlsruhe. *Journal für Mobilität und*

Verkehr, (16):50–58, Mar. 2023. ISSN 2628-4154. doi: 10.34647/jmv.nr16.id103. URL https://journals.qucosa.de/jmv/article/view/103.

[9] L. Barthelmes, M. E. Görgülü, J. Kübler, M. Kagerbauer, and P. Vortisch. Microscopic Agent-Based Parcel Demand Model for the Simulation of CEP-Based Urban Freight Movements to and from Companies. In U. Clausen and M. Dellbrügge, editors, *Advances in Resilient and Sustainable Transport*, pages 75–92. Springer International Publishing, Cham, 2023. ISBN 978-3-031-28196-9 978-3-031-28236-2. doi: 10.1007/978-3-031-28236-2_6. URL https://link.springer.com/10.1007/978-3-031-28236-2_6. Series Title: Lecture Notes in Logistics.

[10] W. L. Bean and J. W. Joubert. Modelling receiver logistics behaviour. *Procedia Computer Science*, 151:763–768, 2019. ISSN 18770509. doi: 10.1016/j.procs.2019.04.103. URL https://linkinghub.elsevier.com/retrieve/pii/S1877050919305666.

[11] S. Bekhor, C. Dobler, and K. W. Axhausen. Integration of Activity-Based and Agent-Based Models: Case of Tel Aviv, Israel. *Transportation Research Record: Journal of the Transportation Research Board*, 2255(1):38–47, Jan. 2011. ISSN 0361-1981, 2169-4052. doi: 10.3141/2255-05. URL http://journals.sagepub.com/doi/10.3141/2255-05.

[12] BIEK: Bundesverband Paket und Expresslogistik e. V. KEP-Studie 2023 – Analyse des Marktes in Deutschland, 2023. Place: Berlin, Köln.

[13] J. Boerkamps and A. v. Binsbergen. GOODTRIP: A NEW APPROACH FOR MODELLING AND EVALUATING URBAN GOODS DISTRIBUTION. 1999. URL https://api.semanticscholar.org/CorpusID:15059688.

[14] L. Briem, N. Mallig, and P. Vortisch. Creating an integrated agent-based travel demand model by combining mobiTopp and MATSim. *Procedia Computer Science*, 151:776–781, 2019. ISSN 18770509. doi: 10.1016/j.procs.2019.04.105. URL https://linkinghub.elsevier.com/retrieve/pii/S187705091930568X.

[15] J. Bézivin. Model Driven Engineering: An Emerging Technical Space. In R. Lämmel, J. Saraiva, and J. Visser, editors, *Generative and Transformational Techniques in Software Engineering: International Summer School, GTTSE 2005, Braga, Portugal, July 4-8, 2005. Revised Papers*, pages 36–64. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-46235-4. doi: 10.1007/11877028_2. URL https://doi.org/10.1007/11877028_2.

[16] R. A. Cavalcante and M. J. Roorda. Freight Market Interactions Simulation (FREMIS): An Agent-based Modeling Framework. *Procedia Computer Science*, 19:867–873, 2013. ISSN 18770509. doi: 10.1016/j.procs.2013.06.116. URL https://linkinghub.elsevier.com/retrieve/pii/S1877050913007242.

[17] K. Czarnecki and S. Helsen. Feature-based survey of model transformation approaches. *IBM Systems Journal*, 45(3):621–645, 2006. ISSN 0018-8670. doi: 10.1147/sj.453.0621. URL http://ieeexplore.ieee.org/document/5386627/.

[18] G. Dalla Chiara, A. R. Alho, C. Cheng, M. Ben-Akiva, and L. Cheah. Exploring Benefits of Cargo-Cycles versus Trucks for Urban Parcel Delivery under Different Demand Scenarios. *Transportation Research Record: Journal of the Transportation Research Board*, 2674(5):553–562, May 2020. ISSN 0361-1981, 2169-4052. doi: 10.1177/0361198120917162. URL http://journals.sagepub.com/doi/10.1177/0361198120917162.

[19] M. De Bok and L. Tavasszy. An empirical agent-based simulation system for urban goods transport (MASS-GT). *Procedia Computer Science*, 130:126–133, 2018. ISSN 18770509. doi: 10.1016/j.procs.2018.04.021. URL https://linkinghub.elsevier.com/retrieve/pii/S1877050918303715.

[20] D. Di Ruscio, R. Eramo, and A. Pierantonio. Model Transformations. In M. Bernardo, V. Cortellessa, and A. Pierantonio, editors, *Formal Methods for Model-Driven Engineering*, volume 7320, pages 91–136. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-30981-6 978-3-642-30982-3. doi: 10.1007/978-3-642-30982-3_4. URL http://link.springer.com/10.1007/978-3-642-30982-3_4. Series Title: Lecture Notes in Computer Science.

[21] J.-F. Erdelyi, F. Amblard, B. Gaudou, E. Kaddoum, and N. Verstaevel. Exploration of Model Coupling Strategies in a Hybrid Agent-Based Traffic Simulation. In K. H. Van Dam and N. Verstaevel, editors, *Multi-Agent-Based Simulation XXII*, volume 13128, pages 153–167. Springer International Publishing, Cham, 2022. ISBN 978-3-030-94547-3 978-3-030-94548-0. doi: 10.1007/978-3-030-94548-0_12. URL https://link.springer.com/10.1007/978-3-030-94548-0_12. Series Title: Lecture Notes in Computer Science.

[22] eurostat. NACE Rev.2 - Statistical classification of economic activites in the European Community, 2008.

[23] M. J. Fischer, M. L. Outwater, L. L. Cheng, D. N. Ahanotu, and R. Calix. Innovative Framework for Modeling Freight Transportation in Los Angeles County, California.

Transportation Research Record: Journal of the Transportation Research Board, 1906(1):105–112, Jan. 2005. ISSN 0361-1981, 2169-4052. doi: 10.1177/0361198105190600113. URL http://journals.sagepub.com/doi/10.1177/036119810519060011 3.

[24] T. Goldschmidt, S. Becker, and E. Burger. Towards a tool-oriented taxonomy of view-based modelling. In E. J. Sinz and A. Schürr, editors, *Modellierung 2012. Hrsg.: Sinz, Elmar J.. Fachtagung Modellierung, 2012, Bamberg*, 14. - 16. März 2012, Bamberg, pages 59–74. Gesellschaft für Informatik (GI), 2012. ISBN 978-3-88579-295-6.

[25] S. Hidaka, J. Bézivin, Z. Hu, and F. Jouault. Principles and Applications of Model Driven Engineering (1) History and Context of Model Driven Engineering. *Computer Software*, 30, Jan. 2013. doi: 10.11309/jssst.30.3_25.

[26] IATA. IATA - Air Cargo Tariffs and Rules: What You Need to Know, July 2021. URL https://www.iata.org/en/publications/newsletters/iata-knowledge-hub/air-cargo-tariffs-and-rules-what-you-need-to-know/.

[27] H. Klare, M. E. Kramer, M. Langhammer, D. Werle, E. Burger, and R. Reussner. Enabling consistency in view-based system development — The Vitruvius approach. *Journal of Systems and Software*, 171:110815, Jan. 2021. ISSN 01641212. doi: 10.1016/j.jss.2020.110815. URL https://linkinghub.elsevier.com/retrieve/pii/S0164121220302144.

[28] A. G. Kleppe, J. B. Warmer, and W. Bast. *MDA explained: the model driven architecture: practice and promise*. The Addison-Wesley object technology series. Addison-Wesley, Boston, 2003. ISBN 978-0-321-19442-8.

[29] D. S. Kolovos, R. F. Paige, and F. A. C. Polack. Merging Models with the Epsilon Merging Language (EML). In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, O. Nierstrasz, J. Whittle, D. Harel, and G. Reggio, editors, *Model Driven Engineering Languages and Systems*, volume 4199, pages 215–229. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-45772-5 978-3-540-45773-2. doi: 10.1007/11880240_16. URL http://link.springer.com/10.1007/11880240_16. Series Title: Lecture Notes in Computer Science.

[30] J. Kübler, L. Barthelmes, M. E. Görgülü, and A. Reiffer. logiTopp, Mar. 2022. URL https://github.com/kit-ifv/logitopp.

[31] J. Kübler, A. Reiffer, L. Briem, and P. Vortisch. Integrating Neighbours into an Agent-Based Travel Demand Model to Analyse Success Rates of Parcel Deliveries. *Procedia Computer Science*, 201:181–188, 2022. ISSN 18770509. doi: 10.1016/j.procs.2022.03.026. URL https://linkinghub.elsevier.com/retrieve/pii/S1877050922004392.

[32] J. Kübler, L. Barthelmes, M. E. Görgülü, M. Kagerbauer, and P. Vortisch. Modeling Relations Between Companies and CEP Service Providers in an Agent-Based Demand Model using Open-Source Data. *Procedia Computer Science*, 220: 486–494, 2023. ISSN 18770509. doi: 10.1016/j.procs.2023.03.062. URL https://linkinghub.elsevier.com/retrieve/pii/S1877050923005975.

[33] J. Kübler, L. Briem, and N. Mallig. mobiTopp, Sept. 2023. URL https://github.com/kit-ifv/mobitopp. original-date: 2018-05-03T14:12:14Z.

[34] C. Llorca and R. Moeckel. Assesment of the potential of cargo bikes and electrification for last-mile parcel delivery by means of simulation of urban freight flows. *European Transport Research Review*, 13(1):33, Dec. 2021. ISSN 1867-0717, 1866-8887. doi: 10.1186/s12544-021-00491-5. URL https://etrr.springeropen.com/articles/10.1186/s12544-021-00491-5.

[35] N. Mallig and P. Vortisch. Modeling travel demand over a period of one week: The mobiTopp model. 2017. doi: 10.48550/ARXIV.1707.05050. URL https://arxiv.org/abs/1707.05050. Publisher: arXiv Version Number: 1.

[36] N. Mallig, M. Kagerbauer, and P. Vortisch. mobiTopp – A Modular Agent-based Travel Demand Modelling Framework. *Procedia Computer Science*, 19:854–859, 2013. ISSN 18770509. doi: 10.1016/j.procs.2013.06.114. URL https://linkinghub.elsevier.com/retrieve/pii/S1877050913007229.

[37] E. Marcucci, V. Gatta, and M. L. Pira, editors. *Handbook on city logistics and urban freight*. Research handbooks in transport studies series. Edward Elgar Publishing, Cheltenham, UK ; Northampton, MA, 2023. ISBN 978-1-80037-016-6.

[38] T. Mens and P. Van Gorp. A Taxonomy of Model Transformation. *Electronic Notes in Theoretical Computer Science*, 152:125–142, Mar. 2006. ISSN 15710661. doi: 10.1016/j.entcs.2005.10.021. URL https://linkinghub.elsevier.com/retrieve/pii/S1571066106001435.

[39] A. Nash, D. Huerlimann, J. Schütte, and V. P. Krauss. *Railml - a standard data interface for railroad applications*, volume 74. WIT Press, 2004.

[40] S. Nolte. *QVT - Relations Language: Modellierung mit der Query Views Transformation*. Xpert.press. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-92170-7 978-3-540-92171-4. doi: 10.1007/978-3-540-92171-4. URL https://link.springer.com/10.1007/978-3-540-92171-4.

[41] A. Nuzzolo and A. Comi. Urban freight demand forecasting: A mixed quantity/delivery/vehicle-based model. *Transportation Research Part E: Logistics and Transportation Review*, 65:84–98, May 2014. ISSN 13665545. doi: 10.1016/j.tre.2013.12.014. URL https://linkinghub.elsevier.com/retrieve/pii/S1366554513002160.

[42] O. M. G. (OMG). Object Constraint Language, v2.4, Feb. 2014. URL https://www.omg.org/spec/OCL/2.4.

[43] O. M. G. (OMG). Meta Object Facility, v2.5.1, Oct. 2016. URL https://www.omg.org/spec/MOF/2.5.1.

[44] O. M. G. (OMG). MOF Query/View/Transformation, v1.3, June 2016. URL https://www.omg.org/spec/QVT/1.3.

[45] O. M. G. (OMG). Unified Modeling Language, v2.5.1, Dec. 2017. URL https://www.omg.org/spec/UML/2.5.1.

[46] D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, Aug. 2007. ISSN 03050548. doi: 10.1016/j.cor.2005.09.012. URL https://linkinghub.elsevier.com/retrieve/pii/S0305054805003023.

[47] K. Pohl, G. Böckle, and F. Van Der Linden. *Software Product Line Engineering*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-24372-4 978-3-540-28901-2. doi: 10.1007/3-540-28901-1. URL http://link.springer.com/10.1007/3-540-28901-1.

[48] E. Popovici, A. Bucci, R. P. Wiegand, and E. D. De Jong. Coevolutionary Principles. In G. Rozenberg, T. Bäck, and J. N. Kok, editors, *Handbook of Natural Computing*, pages 987–1033. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-540-92909-3 978-3-540-92910-9. doi: 10.1007/978-3-540-92910-9_31. URL https://link.springer.com/10.1007/978-3-540-92910-9_31.

[49] A. Reiffer, J. Kübler, L. Briem, M. Kagerbauer, and P. Vortisch. Integrating Urban Last-Mile Package Deliveries into an Agent-Based Travel Demand Model. *Procedia*

Computer Science, 184:178–185, 2021. ISSN 18770509. doi: 10.1016/j.procs.2021.03.028. URL https://linkinghub.elsevier.com/retrieve/pii/S1877050921006542.

[50] M. J. Roorda, R. Cavalcante, S. McCabe, and H. Kwan. A conceptual framework for agent-based modelling of logistics services. *Transportation Research Part E: Logistics and Transportation Review*, 46(1):18–31, Jan. 2010. ISSN 13665545. doi: 10.1016/j.tre.2009.06.002. URL https://linkinghub.elsevier.com/retrieve/pii/S1366554509000817.

[51] T. Sakai, A. Romano Alho, B. Bhavathrathan, G. D. Chiara, R. Gopalakrishnan, P. Jing, T. Hyodo, L. Cheah, and M. Ben-Akiva. SimMobility Freight: An agent-based urban freight simulator for evaluating logistics solutions. *Transportation Research Part E: Logistics and Transportation Review*, 141:102017, Sept. 2020. ISSN 13665545. doi: 10.1016/j.tre.2020.102017. URL https://linkinghub.elsevier.com/retrieve/pii/S1366554520306682.

[52] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck. Record Breaking Optimization Results Using the Ruin and Recreate Principle. *Journal of Computational Physics*, 159(2):139–171, Apr. 2000. ISSN 00219991. doi: 10.1006/jcph.1999.6413. URL https://linkinghub.elsevier.com/retrieve/pii/S0021999199964136.

[53] S. Schroeder, M. Zilske, G. Liedtke, and K. Nagel. Towards a Multi-Agent Logistics and Commercial Transport Model: The Transport Service Provider's View. *Procedia - Social and Behavioral Sciences*, 39:649–663, 2012. ISSN 18770428. doi: 10.1016/j.sbspro.2012.03.137. URL https://linkinghub.elsevier.com/retrieve/pii/S1877042812006040.

[54] S. Schröder and G. T. Liedtke. Towards an integrated multi-agent urban transport model of passenger and freight. *Research in Transportation Economics*, 64:3–12, Sept. 2017. ISSN 07398859. doi: 10.1016/j.retrec.2016.12.001. URL https://linkinghub.elsevier.com/retrieve/pii/S0739885917303104.

[55] H. Stachowiak. *Allgemeine Modelltheorie*. Springer Verlag, Wien, New York, 1973. URL https://archive.org/details/Stachowiak1973AllgemeineModelltheorie.

[56] T. Stahl, M. Völter, and J. Bettin. *Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management*. dpunkt-Verl, Heidelberg, 1. aufl edition, 2005. ISBN 978-3-89864-310-8.

[57] D. Steinberg. *EMF - Eclipse modeling framework*. Addison-Wesley, Upper Saddle River, NJ., 2. ed. rev. and updated, 2. printing edition, 2011. ISBN 978-0-321-33188-5. OCLC: 712490996.

[58] L. Tavasszy and M. De Bok. Overview of urban freight transport modelling. In E. Marcucci, V. Gatta, and M. Le Pira, editors, *Handbook on City Logistics and Urban Freight*, pages 60–77. Edward Elgar Publishing, June 2023. ISBN 978-1-80037-017-3 978-1-80037-016-6. doi: 10.4337/9781800370173.00011. URL https://www.elgaronline.com/view/book/9781800370173/book-part-9781800370173-11.xml.

[59] L. A. Tavasszy and G. d. Jong. *Modelling freight transport*. Elsevier insights. Elsevier, Amsterdam, 1st. edition edition, 2014. ISBN 978-0-12-410400-6.

[60] C. Thaller, B. Dahmen, G. Liedtke, and H. Friedrich. Freight Transport Demand Modelling: Typology for Characterizing Freight Transport Demand Models. In U. Clausen, H. Friedrich, C. Thaller, and C. Geiger, editors, *Commercial Transport*, pages 39–54. Springer International Publishing, Cham, 2016. ISBN 978-3-319-21265-4 978-3-319-21266-1. doi: 10.1007/978-3-319-21266-1_3. URL https://link.springer.com/10.1007/978-3-319-21266-1_3. Series Title: Lecture Notes in Logistics.

[61] TU Berlin, M. Zilske, J. W. Joubert, and University of Pretoria. Freight Traffic. In ETH Zürich, A. Horni, K. Nagel, and TU Berlin, editors, *The Multi-Agent Transport Simulation MATSim*, pages 155–156. Ubiquity Press, Aug. 2016. ISBN 978-1-909188-75-4. doi: 10.5334/baw.24. URL http://www.ubiquitypress.com/site/chapters/10.5334/baw.24/.

[62] M. Zilske, S. Schröder, K. Nagel, and G. Liedtke. Adding freight traffic to MATSim. In *VSP Working Paper*, volume 12-02. TU Berlin, Transport Systems Planning and Transport Telematics, Aug. 2012.