Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 257 (2025) 519–528

Procedia Computer Science

www.elsevier.com/locate/procedia

The 16th International Conference on Ambient Systems, Networks and Technologies (ANT)
April 22-24, 2025, Patras, Greece

Towards Formal Specification of Agent-based Transport Models

Jelle Kübler[a,*], Peter Vortisch[a]

[a]*Institute for Transport Studies, Karlsruhe Institute of Technology (KIT), Kaiserstr. 12, 76131 Karlsruhe, Germany*

Abstract

Agent-Based Transport Models (ABTMs) are powerful tools for simulating complex transport systems, yet they often lack formal specifications, limiting their transparency, reproducibility, and automation. While behavioral (sub-) models such as discrete choice models have a formal foundation, their integration into broader simulation systems is usually informally documented or solely defined in source code, making it difficult to replicate findings, reuse components, or enforce best practices. To address this, this paper conducts a comprehensive analysis of various aspects observed within existing different ABTM frameworks, highlighting the variation of their purposes, resolution, agent types, and modeling methods. Building on this analysis, we propose a metamodel tailored for ABTM specification, defining common concepts in an abstract syntax for defining agent attributes, behaviors, and interactions. Our proposed metamodel emphasizes a method-agnostic design, enabling integration and reuse of already existing behavioral models. This paves the way for automatic model validation, code generation, and sensitivity analysis. Additionally, we outline a conceptual mapping to Discrete Event System Specification (DEVS) to formalize the dynamic semantics, along with a graphical syntax for intuitive model creation by transport engineers.

Keywords: ABM; travel demand; simulation; formal specification; DSL; DEVS

1. Introduction

Agent-based modeling (ABM) is a powerful tool for evaluating the effects of planned infrastructure developments and transport policy measures. In the past decades, this paradigm has gained and retained attention in transport modeling research. It allows to simulate complex, heterogeneous systems with individual agents and their interactions.

Certain aspects of transport modeling already provide well-established formalizations. For example, discrete choice models are based on a stochastic framework, and car-following behavior can be defined through mathematical equations. The composition of such behavioral (sub-)models is rarely, if ever, formally specified. Instead, these integrated agent-based transport models (ABTM) are often described informally using natural language in scientific publications or handbooks. In some cases, their definition is only implicitly encoded in the source code of the simulation itself.

* Corresponding author. Tel.: +49-721-608-42256.
E-mail address: jelle.kuebler@kit.edu

10.1016/j.procs.2025.03.067

This lack of formalization introduces significant challenges. Without a clear, transparent specification, it becomes difficult to identify existing models with desired properties, replicate research findings, or reuse models effectively.

To address these issues, this paper proposes a formal specification language for ABM-based transport models (Sec. 3) by employing concepts from Model-Driven Engineering (MDE) (Sec. 2.1), based on a literature review (Sec. 2.2). Such a language enables new forms of automation, including verifying model properties, enforcing best practices rules, generating and executable code, and identifying interactions among agents to leverage parallel processing. Existing standards like GMNS (General Modeling Network Specification) [7] can be integrated as 'best practice rules' to check that the domain-specification complies with the standard, as well as serialization format for the related aspects of model instances.

Importantly, the specification language must integrate existing (formalized) behavioral (sub-)models. By being 'method-agnostic', it remains independent of the specific methods used to model agent behavior, but focuses on specifying their integration. These behavioral (sub-) models will be considered purely functional models, producing a deterministic output for a given input (including sources of randomness). Existing standards like PMML (Predictive Model Markup Language) [18] can be integrated by incorporating the specified model parameters.

This approach has the potential to increase transparency and therefore trust in ABTM while lowering the technological barriers for transport engineers and scientists (domain experts). Using a domain-specific language (DSL), they can define and engineer models without delving into the underlying complexities of model transformation or code generation, tasks which could instead be managed by technology experts developing DSLs and simulation software.

The term 'model' can have various meanings. In this paper, we specifically use 'model' to refer to ABTMs, which integrate multiple behavioral models to simulate the spatiotemporal evolution of agents within a transport system. We distinguish 'behavioral models' capturing individual choices. They map a given choice situation (including agent data) to a selected choice or value (e.g. a destination, mode of transport, or route). Finally, the terms metamodel and DSL refer to the modeling language proposed in this paper.

2. Related Work and Foundations

There are several popular transport modeling frameworks, some of which touch on topics related to this paper's focus, while most do not fully integrate the principles of Model-Driven Engineering. Bastarianto et al. [5] provide an overview of current ABTM frameworks, based on a literature review spanning the last two decades.

MATSim [22] supports large-scale agent-based transport simulations. It iteratively scores and mutates the daily travel plans of agents (selected modes and routes). During each iteration, a physical simulation module called 'mobsim' is executed, employing a queue-based mesoscopic traffic flow model. MATSim provides nine extension points through the software design pattern of dependency injection [54]. Extensions are implemented in the Java programming language, and the MATSim handbook describes the XML format for input data. Similarly the other listed frameworks: NetLogo [52], SimMobility [1], TAPAS [21], FEATHERS [6] and mobiTopp [30], require engineers to write source code in the respective programming languages and familiarity with the underlying code base to integrate custom behavioral models.

The mobiTopp framework already makes use of domain-specific languages (DSLs) in one aspect: It provides a DSL for specifying discrete choice models (nested logit models) and their utility functions [27], eliminating the need to write custom Java code, as it is automatically generated. Executable Java code is automatically generated from these DSL specifications. This approach makes the choice models more readable and eliminates the Java-specific boilerplate code.

AnyLogic [17], a multimethod modeling environment, provides a graphical model editor. Users can specify agent properties in a tree-like view and processes using flowchart diagrams. However, as proprietary software, AnyLogic's provided libraries (e.g., road traffic) have public API documentation but lack transparency regarding their internal workings. Users can extend the simulations with custom logic written in Java.

Metamodeling has previously been explored in the related domain of Agent-Oriented Software Engineering (AOSE) and multi-agent system (MAS) specification (e.g. SARL [39], ASPECS [12]), where agents are treated as software units. This contrasts with the domain of agent-based simulation (ABS), where agents represent entities in a simulated environment rather than software components. Beydoun et al. [8] provide an extensive overview of ten metamodeling approaches developed for AOSE and MAS specification. To unify these diverse methodologies, they propose a 'methodology-independent' modeling language called FAML that encapsulates shared concepts. However, these AOSE metamodels have limitations when applied to ABTM simulations. First, their terminology and abstraction levels are designed for software and system engineers, not for transport engineers. Second, most of these metamodels are built around the cognitive frameworks like BDI (Belief-Desire-Intention) [37]. Recently, Viana et al. [50] extended a cognitive framework metamodels to model adaptive normative agents. FAML supports but does not mandate the use of such cognitive frameworks. However, FAML and the other listed metamodels showcase a goal- and task-oriented perspective that makes them ill-suited for transport modeling. In transport simulations, the agents' are deigned to reflect the decision-making processes and interactions observed in the real world, often through stochastic models. The global purpose of the simulation is to estimate a future state of a transport system by evaluating scenarios (e.g., infrastructure changes, policy adaptations, or cost evolution). This distinguishes them from software agents (or agent compositions) in terms of AOSE, which model software systems that provide a service - i.e. reacting to client input and providing some output. Another key difference lies in the treatment of the environment. AOSE metamodels typically include concepts to model the environment in which (software) agents are situated and with which they interact (e.g. to model sensors, actuators, or remote data sources). In transport simulations, however, the environment itself is simulated as it is a part of the modeled transport domain. Therefore, it can be modeled as a (potentially static) agent (or set of agents) itself. Lastly, the AOSE metamodels listed by Beydoun et al. [8] often use generalized ontology objects, without specifying how agent attributes are modeled. In the case of AOSE, the domain representation can potentially be implemented manually by users. This, however, is not desired for ABTM specification, as the goal is to remove the need for writing custom source code.

Apart from AOSE, Zhang et al. [53] have combined an agent metamodel for multi-agent simulation with the DEVS (discrete event system specification) framework. However, their approach focused on mapping the cognitive framework PRS onto DEVS concepts and is therefore 'method-dependent'. Transport simulations often rely on statistical regression models such as discrete choice models, making their approach non-applicable. This further underscores the need for metamodeling approaches tailored to the unique requirements of agent-based transport simulations.

Furthermore, the co-simulation framework MECSYCO (Multi-Environment Co-Simulation and Modeling Framework) [43, 10], allows to integrate multiple simulation models across various domains. The interaction of different models, or "simulators", can be defined by specifying data exchange interfaces and coupling artifacts (supporting the FMU standard). The framework's metamodel is mapped to the DEVS formalism for consistency and synchronization across the co-simulation. However, agents in the MECSYCO metamodel represent entire simulators rather than individual entities. The abstraction through artifacts obscures the domain models as well as the side effects (updates of state variables) inside these black-box simulators.

2.1. Model-Driven Engineering

Model-Driven Development (MDD) is a paradigm aiming to reduce accidental complexity (opposed to essential complexity inherent within the application domain) in system development by using models as central artifacts throughout the engineering life-cycle [2]. Historically, software models were often limited to design documentation or high-level system descriptions, not used in essential phases such as testing, deployment, or runtime adaptation. Hence, in conventional software engineering, models are second-class entities, disconnected from the software artifacts they represented. MDD raises models to become first-class objects in software development. Models are considered as an integral artifacts in the engineering process enabling automation like code generation. Furthermore, MDD defines formal, machine-readable formats with well-defined semantics, making models interoperable and reusable [2].

Model-Driven Engineering (MDE), employs this MDD paradigm: technology experts develop domain-specific languages (DSL) (e.g. a language to specify ABM), that allows domain-experts (e.g. transport engineers) to specify their system at an abstract level without knowledge of programming languages or software architecture. MDE applies model transformations and code generation (algorithms defined by the technology expert, i.e. DSL software developer) to create executable code or perform analyses of the specified models [42].

The general model theory by Stachowiak [44] defines the term 'model' through three features: A model represents something called the 'original' (representation), it captures only certain attributes of the original (reduction), and it serves a specific purpose (pragmatic).

To develop a formal DSL for a certain domain, a metamodel can be developed, which defines the abstract elements, structure and semantics of the models defined by the domain experts. This defines the bounds in which to create models of their system [3, 45]. A metamodel consists of four aspects: The *abstract syntax* describes the potential structure of models. It defines the constructs of a DSL and the relationships between those constructs. The *concrete syntax* determines how these constructs are visually or textually represented (e.g. JSON format, or graphical diagrams) [26]. *Context conditions* describe additional modeling rules (constraints) that cannot be expressed by the abstract syntax [19] (e.g. OCL [51]). The *dynamic semantics* of a metamodel defines the meaning of its constructs and their relationships. These semantics are often defined informally in natural language text. Alternatively the metamodel can be mapped to a language with defined semantics (e.g. Petri-networks, or programming languages)[14].

To formalize the creation of metamodels, a (self-describing) meta-metamodel was developed: the Meta-Object Facility (MOF) [35]. The defacto reference implementation of the subset ’essential MOF’ (EMOF) is ECore provided by the Eclipse Modeling Framework (EMF) [46]. Figure 1 shows the four modeling layers with exemplary ’instance-of’ relations.

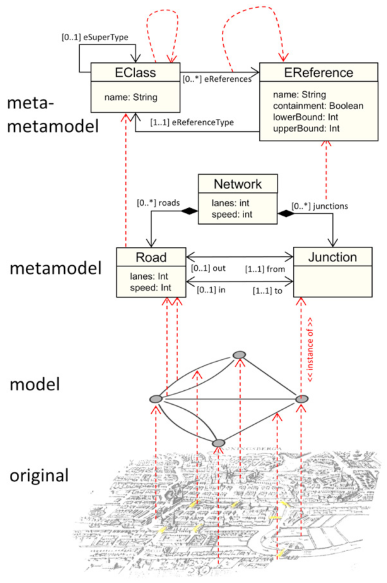

Fig. 1: Modeling layers with exemplary ’instance’ relations: an original city (image based on [34]), a graph model of the road network, a metamodel defining the structure of graphs, and a meta-metamodel (ECORE) describing the structure of the metamodel.

2.2. Dimensions of Variety of Agent-based Transport Models

Transport demand models often concentrate on modeling the transport of either people or goods. A second dimension of classification is whether trips are conducted privately or commercially, resulting in: private passenger transport (trips to shop, leisure, work, home), business passenger transport (business trips conducted during work), commercial freight transport, and private freight transport (private deliveries or relocation). Models of private travel demand have been the focus of several studies and developed frameworks. In recent years, models of freight transport gained more attention; some demand models even integrate both private passenger and freight transport to account for their interactions [25, 4, 38, 40, 55, 48]. Even within these ”classes” of transport models, there is still high variation in which aspects of behavior are modeled (most models include multiple aspects). For private passengers, this includes the choice of: a place of work or education, ownership of mobility tools, activities and their scheduling, or destination, mode of transportation and route for each trip. Freight demand models model a different set of decisions, including: establishment location, fleet size, contracts with carriers, the demand for and production of goods, and tour planning. Meanwhile, traffic flow models generally distinguish vehicle classes such as passenger cars or heavy vehicles, thus abstracting the trip purposes inherent within these four classes. Furthermore, the different modeled aspects of transportation are sometimes distinguished into three levels: strategic, tactical, or operational. Moreover, through its pragmatic feature, each transport model has a certain purpose or certain operations for which it is built. Models are usually built to assess current or planned transport infrastructure or to forecast a future state of the transport system, after implementing certain measures (like building infrastructure or employing transport policies). A model’s purpose defines which aspects it should reflect and which of those must be sensitive to the implemented measures/policies.

ABTM also vary in their spatial and temporal scope, depending on the focus and objectives of the simulation. The spatial scope can range from analyzing single intersections or highway sections (e.g. traffic flow models), to entire cities, metropolitan areas, or countries (e.g. travel demand or land-use models) [24]. The categorization of time scales in transport models varies across studies and depends on both the use case and the modeled aspects of transportation. The often used terms of short-term, mid-term, and long-term models are ambiguous: Huang et al. [24] define short-term models as within-day microscopic agent behavior (e.g., lane-changing and collision avoidance); mid-term as day-to-day decisions like scheduling, route- and mode choice; and long-term as decisions spanning months to years (land use or property development). Lu et al. [29] use a similar definition for the different time scales within the SimMobility framework. In contrast, mobiTopp uses a week-long perspective, defining long-term as invariant decisions over the simulated week, while short-term covers within-day decisions [30, 20]. Other studies define ”long-term” as up to multiple decades [47, 36, 9], while studies in flow prediction define it as above 15-30 minutes [49] or 1 hour [28].

The next dimension of variety is the type(s) of simulated agents. Huang et al. [24] discuss eight properties for ABTM based on DeLaurentis' [13] definition for aeronautics: autonomous, interface, mobile, adaptive, reactive, utility-based, goal-based and information-gathering agents. We will exclude the 'autonomous'/'interface' agent dimension, as we assume fully autonomous models, that require no additional user input at runtime other than the provided input data. This leaves us with the mobility dimension (mobile vs. immobile agents) and the paradigm of decision making: either utility-based or goal-based. Furthermore, reactive behavior describes non-learning agents (with limited or no memory and a fixed solution path) while adaptive agents change their behavior based on past experiences (and hence require memory) (cf. [24]). Finally, we identify the 'information-gathering' property as a prerequisite for an agent to be adaptive. Specifying these properties of an agent simplifies defining behavioral rules in later steps.

Another aspect by which transport models are commonly classified is their resolution. Typically, micro-, meso- and macroscopic models are distinguished: microscopic refers to models with individual entities and their interactions; mesoscopic models aggregate individuals with (some) shared attributes (like current road link or destination); macroscopic models aggregate (large portions of) the population through a few key figures [5, 16]. However, for integrated ABTM that model multiple decisions, this classification can be ambiguous: for example, the MATSim framework models individual person agents with their own attributes and plans [23], hence its travel demand component is microscopic. However, MATSim is also mesoscopic in that it provides a queue-based model of traffic flow [22]. Also, the agent-based travel demand modeling framework mobiTopp [30] simulates individual schedules of microscopic agents (incl. destination and mode choice), however, excludes rout choice resulting in aggregated macroscopic origin-destination matrices. Handling (millions of) agents on a microscopic level (especially for large-scale scenarios/ survey areas) results in high computational effort in both computation power and memory [24, 5]. Therefore, Manley et al. [31] developed a hybrid approach to balance the trade-off between realistic behavior and computational effort. Also, Mathieu et al. [32] argue that multi-agent systems should incorporate a ”ZOOM” pattern, to dynamically switch between a micro- and macroscopic perspective where possible.

Lastly, transport models can vary in their applied method(s) to model certain behavioral aspects. A common method is random utility maximization, where agents aim to optimize outcomes such as minimizing travel time or costs [33]. while Mehdizadeh et al. [33] also list random regret minimization [11] as a relevant non-utilitarian discrete choice model alternative. Moreover, cognitive frameworks like the Belief-Desire-Intention [37] paradigm or PECS (Physis, Emotion, Cognition, Social Status) [41] can be applied. Mehdizadeh et al. [33] recommend using theory-driven models over heuristic models as their established behavioral frameworks provide more straightforward causal links.

3. Proposed Metamodel

Our proposed metamodel is structured into three core components: the static description of agents and their properties (data model), the description of dynamic agent behavior and interactions (process model), and the specification of behavioral model interfaces (extension points). First, we will define its core concepts and constraints (abstract syntax and context conditions), based on the reviewed literature of Section 2. Later, we propose first ideas for a graphical, diagram-based syntax as well as a sketch for mapping the dynamic part of the metamodel to the DEVS formalism to define its semantics. We choose the mapping to DEVS as this has been found effective for transport models in previous research [15], however, a mapping to other standards like SARL [39] are conceivable and equally effective.

The root element *AgentBasedTransportModel* of our metamodel holds the following agent definitions and some metadata properties like a name, coordinate system, and spatial bounds of the modeled area. As discussed in Section 2.2, conventional terms used for time-scale and resolution classification can be ambiguous for integrated ABTM. Therefore, we do not specify them for the entire ABTM globally, but instead specify them per sub-model.

The root concept of our metamodel is 'AgentBasedTransportModel' which contains the following model elements and holds some additional metadata. Each model has a name, coordinate system, and spatial bounds of the modeled area. Some metadata attributes, like spatial bounds, could even be determined automatically (e.g., by computing the convex hull of the other elements). As discussed in Section **??**, the terms such as short-, mid-, and long-term models are ambiguous. Instead, users should define the (ABTM-wide) simulation period (interval) and the expected time scale for each behavioral model. With this, we can derive the expected temporal resolution of certain decisions. Similarly,

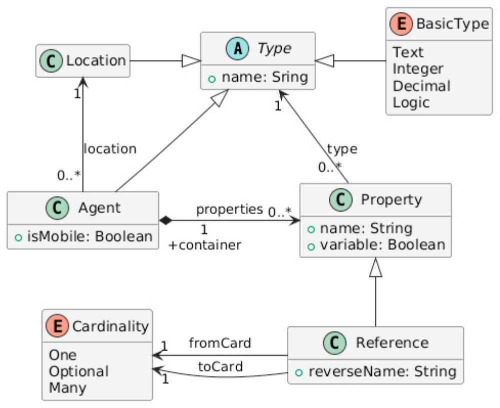

Fig. 2: Metamodel to specify the static agent data model. UML-Labels: (A)bstract, (C)lass, (E)numeration.

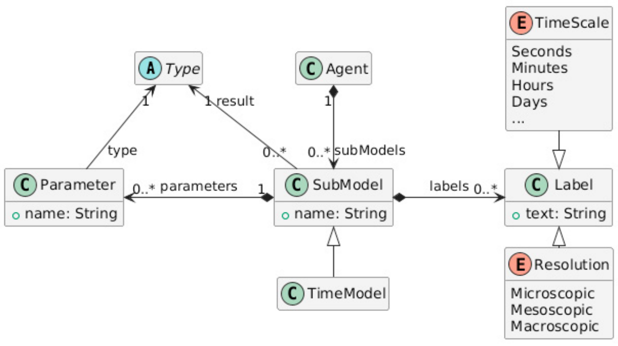

Fig. 3: Metamodel to specify (behavioral) sub-models

the conventional classification of models as micro-, meso-, or macroscopic is avoided at the ABTM level; instead, the agent-resolution may also be specified individually for each behavioral model.

The static agent data model (Fig. 2) describes types of *Agents* and their *Properties*. Each type of *Agent* is given a *name* and categorized as either *mobile* or stationary. Every *Agent* has a spatial embedding (*Location*, e.g. point, polyline, or polygon). For *mobile* agents, this embedding is variable during simulation, while for stationary agents, it remains immutable. Furthermore, *Agents* possess a set of *Properties*, each defined by a *name* and *type*. Basic types include *Text*, *Logical*, *Integer* and *Decimal*. A property can also *Reference* other *Agents* (so they are themselves *Types*), either as a single property (e.g., a person's household) or as a set *Reference* (e.g., all members of a household). *References* also have a *reverse name* to model how the referenced *Agent* calls the referencing *Agent*. *Cardinality* constraints express whether properties are optional or can hold lists of values. Advanced data structures, such as *Schedule*, *Route*, or *Network*, may also be included in the future as additional, frequently used *Types*. *Properties* are classified as either static (immutable) or state *variables* that evolve during the simulation. State *variables* capture an agent's internal state or memory and can be used to model adaptive behavior, stability variables in utility functions, or cognitive frameworks such as BDI (Sec. 2.2).

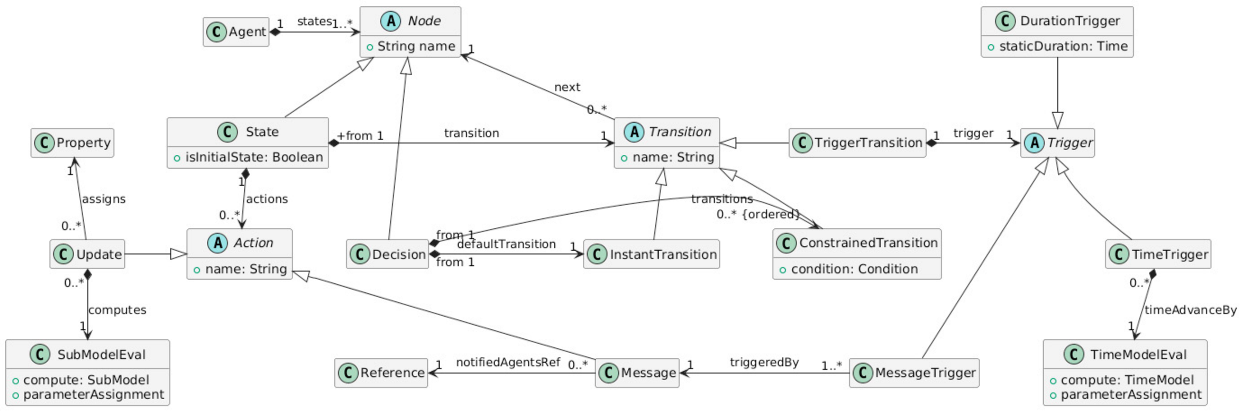

Fig. 4: Metamodel of agent dynamics and interactions

To achieve a method-agnostic metamodel of ABTM, our approach decouples the specification of behavioral models (here called *Sub-Models* to also include other types of isolated model procedures) from the integrated ABTM specification (Fig. 3). *Sub-Models* represent pure functions that map a (choice) situation (input *parameters* such as: agent attributes, current state and location) to an outcome (*result type* such as: destination, mode, or route). This approach encourages to reuse existing (behavioral) models which often already have a high degree of formalization. *Sub-Models* can have multiple implementations, enabling contributions from both technology experts (e.g., route search algorithms) and domain experts (e.g., discrete choice models). *Sub-Models* may also include *Labels* (a *Resolution*, *Time Scale* or custom text) so users can convey their intended application, such as 'utility-based', 'goal-based' or 'information-gathering' as discussed in Section 2.2. In the literature these classifications are attributed to *Agents*, however, as *Agent* behavior can be described through multiple *Sub-Models*, we apply these *Labels* per *Sub-Model*.

The dynamic model (Fig. 4) integrates the *Agent* data model and behavioral *Sub-Models* to describe the evolution of *Agent States* and interactions. *Agent* behavior can depend on its current situation like 'at home', 'in transit' or 'overtaking process'. To explicitly show the differences in behavior, *States* and *Transitions* can be modeled. *States* have a *Transition* to either the *next State* or *Decision* node. *Decision* nodes can have multiple (ordered) *Constrained Transitions* additional to a *default transition*. Exactly one *State* must be marked as initial. At the entry of a *State*, (behavioral) *Sub-Models* can be applied to update the *Agent*'s state *variables*. These *Updates* are explicitly defined, to transparently state the 'side-effects'. Here, the *Type* of the assigned *Property* must match that of the *computed Sub-Model*. State *Transitions* can be *instant* or triggered by elapsed time (*Time Trigger*), or external messages (*Message Trigger*). The time for state *Transitions* is either a static *Duration* or modeled dynamically by applying a *Time-Model* with a *result type* of *Time*. Durations can be 'infinite' to model passive states that transition only in response to external messages. *Messages* provide a mechanism for *Agent* interactions in situations where state progression of one *Agent* depends on another. When entering a *State*, *Agents* can send *Messages* to other *referenced Agents*. For example, a vehicle agent may notify a person agent with an 'arrived' message, enabling the person to begin their next activity.

The dynamic semantics of the proposed metamodel can be defined as a mapping to the Discrete Event System Specification (DEVS) formalism. The following sketch shows the feasibility of such a mapping. A complete mapping is part of our future work. The static aspects of the metamodel, i.e. the agent data model and the behavioral model interface descriptions we can adopt semantics similar to UML or Ecore class diagrams. One crucial aspect, however, is the visibility and mutability of agent state variables: we assume only the agent itself can modify its variable properties. This encapsulation ensures that state changes are explicit and controlled within the agent's behavior.

The dynamic aspects of the metamodel, specifying the agent behavior and interactions, require a more specific mapping to DEVS. Previous research, such as the work of Dubiel and Tsimhoni [15], showed that combining agent-based modeling (ABM) with DEVS enhances the capabilities of both approaches. Moreover, such a mapping also enables the generation of executable simulation code, as there are numerous existing DEVS-based simulation engines.

A DEVS model is formally defined as $M = (X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta)$ where:

- S: Set of system states; X: Set of input events; Y: Set of output events.
- $\delta_{ext} : S \times X \times \mathbb{R}_0^+ \rightarrow S$: External transition function: determines next state based on current state, input, and elapsed time since the last state transition.
- $\delta_{int} : S \rightarrow S$: Internal transition function: determines next state when no external event occurs.
- $\lambda : S \rightarrow Y$: Output function: generates the output events just before an internal transition.
- $ta : S \rightarrow \mathbb{R}_0^+ \cup \{\infty\}$: Time advance function: determines the duration a model remains in a given state.

Each agent A defined in the proposed metamodel can be represented by a corresponding DEVS model M_A. A's states directly map to the M_A's state set S. Input events X in M_A correspond to the messages A expects during its message-triggered transitions, while the output events Y represent the messages sent by A. The DEVS output function λ (executed before internal transition) can be defined to execute the update actions and return the sent messages of the current transition's next state.

A's state transitions $t := s \rightarrow s'$ translate to M_A's transition functions. Instant transition have a time advance of 0. For time-triggered transitions ta determines the duration A remains in state s by evaluating the associated time model. For message-triggered transitions, ta returns infinity, making s a passive state in the DEVS framework. The external transition function δ_{ext} handles state changes based on incoming messages. If a message x aligns with transition t's trigger, the external transition function returns the next state s'. Otherwise, the agent remains in its current state s. When M_A receives a message not expected during transition t, a special 'REJECT' response can be sent back. A decision node d with multiple constrained transitions is mapped to a state s_d with $ta = 0$, hence the conditions are evaluated as soon as s_d is entered. $\delta_{int}(s_d)$ evaluates the conditions in the specified order, selecting the first valid transition or the default transition if none match. This also supports probabilistic state transitions: by incorporating random variables into transition conditions.

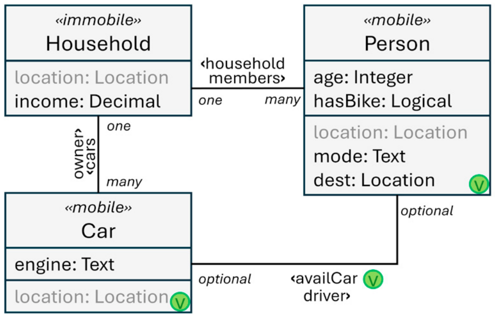

Fig. 5: Example agent data model: persons, households and cars with static and variable (v) properties and references.

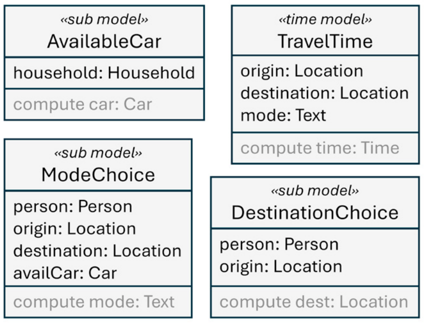

Fig. 6: Example behavioral models: mode and destination choice, travel time model and car availability check, each without labels.

Figures 5,6 and 7 show a draft how an exemplary ABTM could be defined using a graphical, diagram based syntax. The example shows person and car agents assigned to households. Each person agent executes activities (here one hour) before planning a trip, i.e. selecting a destination, checking car availability and selecting a mode. The duration is determined by the current location, selected destination and mode. In case the mode ’CAR’ is selected, the available car is notified about the trip start (and end). A ’fail-safe’ mechanism handles unexpected ’REJECT’ messages of the car. To define the static agents data model, we propose to use a simplified version of UML or ECore class diagram syntax (Fig. 5), and an extended UML state diagram syntax for the dynamic model (Fig. 7).

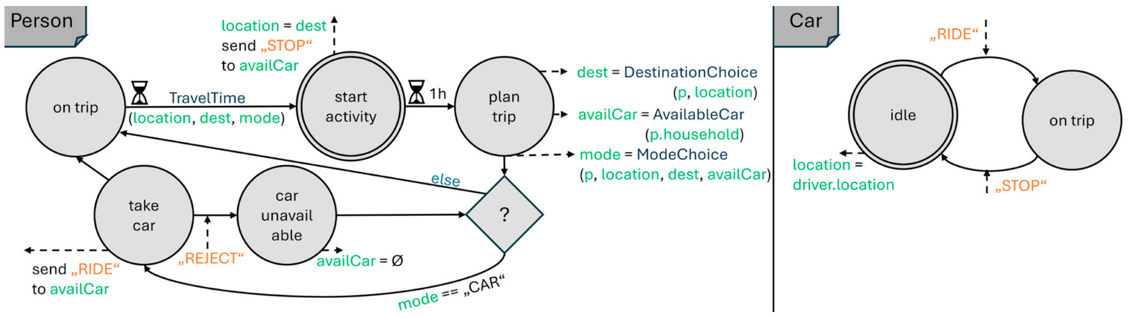

Fig. 7: Example dynamic models of Person and Car. Notation: circle = state (initial: double circle); diamond = decision node; solid arrows = transition; hourglass = duration/time trigger; inbound dashed arrow = message trigger; outbound dashed arrow = update/send message.

4. Conclusion

In this paper, we first analyzed various aspects of agent-based transport models (ABTMs) as described in the literature. Building on this analysis, we proposed a metamodel for specifying ABTMs. This metamodel includes an abstract syntax for modeling agents and their attributes, as well as their dynamic behavior and interactions. A key feature of our approach is its method-agnostic design concerning (behavioral) sub-models (individual decisions without updating the agent’s state). This approach enables the reuse of previously developed models, such as discrete choice models or routing algorithms. To support this, the metamodel allows to define interfaces for these sub-models. To define the dynamic semantics of a specified ABTM, we provided a conceptual mapping to the DEVS formalism. This mapping not only establishes a formal definition of semantics but also a transformation into executable simulation code. Future research could explore additional mappings to integrate existing frameworks like MATSim, enabling reuse of their mobsim component. Finally, we proposed an initial draft of a graphical, diagram-based concrete syntax that transport engineers can use to create ABTMs.

With the developed domain specific language for ABTM specification, we take a step towards more transparent and formalized (integrated) ABTM. The formal specification allows for more automation in the design/engineering process. Best practices could be formulated with reference to the proposed metamodel and later automatically checked or enforced. Moreover, a formally specified model could be automatically checked to determine if it will be sensitive to certain changes like increased cost travel time or automated vehicle saturation. Here, the question remains open, whether a list of (behavioral) sub-models suffices to capture the purpose of an ABTM. In future work, we will investigate if desired model sensitivities could be specified as well.

To fully unlock the potential of automation and automated reasoning about ABTM, the specified behavioral (sub-) model interfaces could be extended by including formal contracts defining pre- and post-conditions on in- and outputs. To further increase the reuse of these sub-models, a concept similar to generalization (or inheritance) in UML could be integrated to the static agent data models, hence allowing for adding custom properties while previously developed sub-models still operate in the more general view of the parent class.

Some additional extensions of the metamodel are conceivable, to make it more expressive like: constraining the domain of property values, adding custom role labels to indicate the purpose of properties, multiple result values for sub-models to reduce redundancy, or combining optional and mandatory transition triggers. Another promising future extension is the incorporation of model validation and calibration similar to Beydoun et al. [8] who proposed to include a design-time view.

Acknowledgements

This work was conducted within the PoF program 46.24.01 - Applied TA: Digitalization and Automation in the Context of Socio-Technical Change.

References

[1] Adnan, M., Pereira, F., Lima Azevedo, C., Basak, K., Lovric, M., Raveau, S., Zhu, Y., Ferreira, J., Zegras, C., Ben-Akiva, M., 2016. SimMobility: A Multi-Scale Integrated Agent-based Simulation Platform.

[2] Atkinson, C., Kuhne, T., 2003. Model-driven development: a metamodeling foundation. IEEE Software 20, 36–41.

[3] Balsam, M., Mazlina, M., 2022. The Role of Metamodeling in Systems Development, pp. 2421–2436.

[4] Barthelmes, L., Görgülü, M.E., Kübler, J., Kagerbauer, M., Vortisch, P., 2023. Microscopic Agent-Based Parcel Demand Model for the Simulation of CEP-Based Urban Freight Movements to and from Companies, in: Clausen, U., Dellbrügge, M. (Eds.), Advances in Resilient and Sustainable Transport, Springer International Publishing, Cham. pp. 75–92.

[5] Bastarianto, F.F., Hancock, T.O., Choudhury, C.F., Manley, E., 2023. Agent-based models in urban transportation: review, challenges, and opportunities. European Transport Research Review 15, 19.

[6] Bellemans, T., Kochan, B., Janssens, D., Wets, G., Arentze, T., Timmermans, H., 2010. Implementation Framework and Development Trajectory of FEATHERS Activity-Based Simulation Platform. Journal of the Transportation Research Board 2175.

[7] Berg, I., Smith, S., Zhou, X.S., 2022. GMNS: A Specification for Sharing Routable Road Networks. Technical Report. URL: `https://trid.trb.org/View/1909441`.

[8] Beydoun, G., Low, G., Henderson-Sellers, B., Mouratidis, H., Gomez-Sanz, J.J., Pavon, J., Gonzalez-Perez, C., 2009. FAML: A Generic Metamodel for MAS Development. IEEE Transactions on Software Engineering 35, 841–863.

[9] Blainey, S.P., 2012. A long term capacity and demand assessment model for the UK transport system .

[10] Camus, B., Vaubourg, J., Presse, Y., Elvinger, V., Paris, T., Tan, A., Chevrier, V., Ciarletta, L., Bourjot, C., de Lorraine, U., Umr, L., 2016. Multi-agent Environment for Complex SYstems COsimulation (MECSYCO) - Architecture Documentation.

[11] Chorus, C.G., 2012. Random Regret-based discrete choice modeling: A tutorial .

[12] Cossentino, M., Gaud, N., Hilaire, V., Galland, S., Koukam, A., 2010. ASPECS: An agent-oriented software process for engineering complex systems. Autonomous Agents and Multi-Agent Systems 20, 260–304.

[13] DeLaurentis, D., 2005. Understanding Transportation as a System-of-Systems Design Problem, in: 43rd AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics, Reno, Nevada.

[14] Domínguez, E., Zapata, M.A., 2000. Mappings and Interoperability: A Meta—modelling Approach, in: Advances in Information Systems. Springer Berlin Heidelberg, Berlin, Heidelberg. volume 1909, pp. 352–362.

[15] Dubiel, B., Tsimhoni, O., 2005. Integrating agent based modeling into a discrete event simulation, in: Proceedings of the Winter Simulation Conference, 2005.

[16] Farhana, Y., Morency, C., Roorda, M.J., 2017. Macro-, meso-, and micro-level validation of an activity-based travel demand model. Transportmetrica A: Transport Science 13, 222–249.

[17] Grigoryev, I., 2024. AnyLogic in Three Days - A quick course in simulation modeling. URL: https://www.anylogic.com/upload/al-in-3-days/anylogic-in-3-days.pdf.

[18] Guazzelli, A., Zeller, M., Lin, W.C., Williams, G., 2009. PMML: An Open Standard for Sharing Models. The R Journal 1, 60.

[19] Harel, D., Rumpe, B., 2004. Meaningful modeling: what's the semantics of "semantics"? Computer 37, 64–72.

[20] Hilgert, T., Heilig, M., Kagerbauer, M., Vortisch, P., 2017. Modeling Week Activity Schedules for Travel Demand Models. Journal of the Transportation Research Board 2666, 69–77.

[21] Holmgren, J., Davidsson, P., Persson, J.A., Ramstedt, L., 2012. TAPAS: A multi-agent-based model for simulation of transport chains. Simulation Modelling Practice and Theory 23, 1–18.

[22] Horni, A., Nagel, K., Axhausen, K.W. (Eds.), 2016. The Multi-Agent Transport Simulation MATSim. Ubiquity Press.

[23] Horni, A., Nagel, K., Axhausen, K.W., 2024. MATSim User Guide. The Multi-Agent Transport Simulation MATSim.

[24] Huang, J., Cui, Y., Zhang, L., Tong, W., Shi, Y., Liu, Z., 2022. An Overview of Agent-Based Models for Transport Simulation and Analysis. Journal of Advanced Transportation 2022.

[25] de Jong, G., de Bok, M., Thoen, S., 2021. Seven fat years or seven lean years for freight transport modelling? Developments since 2013. Journal of Transport Economics and Policy (JTEP) 55, 124–140.

[26] Krahn, H., Rumpe, B., Völkel, S., 2008. MontiCore: Modular Development of Textual Domain Specific Languages, in: Objects, Components, Models and Patterns. Springer Berlin Heidelberg, Berlin, Heidelberg. volume 11, pp. 297–315.

[27] Kübler, J., 2023. mobiTopp-rastatt - config/choice-models. URL: https://github.com/kit-ifv/mobitopp-rastatt.

[28] Lin, H., Dai, H., Tseng, V., 2024. Short- and Long-Term Travel Time Prediction Using Transformer-Based Techniques. Applied Sciences 14.

[29] Lu, Y., Adnan, M., Basak, K., Pereira, F., C., C., V.H., S., H., L., Ben-Akiva, M., 2015. SimMobility Mid-Term Simulator: A State of the Art Integrated Agent Based Demand and Supply Model, in: 94th Annual Meeting of the Transportation Research Board, Washington, DC.

[30] Mallig, N., Vortisch, P., 2017. Modeling travel demand over a period of one week: The mobiTopp model.

[31] Manley, E., Cheng, T., Penn, A., Emmonds, A., 2014. A framework for simulating large-scale complex urban traffic dynamics through hybrid agent-based modelling. Computers, Environment and Urban Systems 44, 27–36.

[32] Mathieu, P., Morvan, G., Picault, S., 2018. Multi-level agent-based simulations: Four design patterns. Simulation Modelling Practice and Theory 83, 51–64.

[33] Mehdizadeh, M., Nordfjaern, T., Klöckner, C.A., 2022. A systematic review of the agent-based modelling/simulation paradigm in mobility transition. Technological Forecasting and Social Change 184, 122011.

[34] Merian-Erben, 100. Königsberg 1651. URL: https://commons.wikimedia.org/w/index.php?curid=1447648.

[35] (OMG), O.M.G., 2016. Meta Object Facility (MOF). URL: https://www.omg.org/spec/MOF.

[36] Peeters, P.M., 2013. Developing a long-term global tourism transport model using a behavioural approach: implications for sustainable tourism policy making. Journal of Sustainable Tourism 21, 1049–1069.

[37] Rao, A.S., Georgeff, M.P., 1995. BDI agents: from theory to practice., in: Icmas, pp. 312–319.

[38] Reiffer, A., Kübler, J., Briem, L., Kagerbauer, M., Vortisch, P., 2021. An integrated agent-based model of travel demand and package deliveries. Journal of Traffic and Transportation Management 3, 17–24.

[39] Rodriguez, S., Gaud, N., Galland, S., 2014. SARL: A General-Purpose Agent-Oriented Programming Language.

[40] Sakai, T., Alho, A.R., Bhavathrathan, B.K., Dalla Chiara, G., Gopalakrishnan, R., Jing, P., Hyodo, T., Cheah, L., Ben-Akiva, M., 2020. SimMobility Freight: An agent-based urban freight simulator for evaluating logistics solutions. Transportation Research Part E: Logistics and Transportation Review 141, 102017.

[41] Schmidt, B., 2000. The modelling of human behaviour: The PECS reference models. SCS-Europe BVBA Delft.

[42] Schmidt, D., 2006. Guest Editor's Introduction: Model-Driven Engineering. Computer 39, 25–31.

[43] Siebert, J., Ciarletta, L., Chevrier, V., 2010. Agents and artefacts for multiple models co-evolution. Building complex system simulation as a set of interacting models, in: Autonomous Agents and Multiagent Systems-AAMAS 2010, ACM. pp. 509–516.

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

[45] Stahl, T., Efftinge, S., Haase, A., Völter, M., 2012. Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management. dpunkt.

[46] Steinberg, D., Budinsky, F., Paternostro, M., Merks, E., 2008. Eclipse Modeling Framework. Second edition ed., Pearson Education.

[47] Stephenson, J., Zheng, L., . National long-term land transport demand model September 2013 .

[48] Stinson, M., Mohammadian, A.K., 2022. Introducing CRISTAL: A model of collaborative, informed, strategic trade agents with logistics. Transportation Research Interdisciplinary Perspectives 13, 100539.

[49] Toqué, F., Khouadjia, M., Come, E., Trepanier, M., Oukhellou, L., 2017. Short & long term forecasting of multimodal transport passenger flows with machine learning methods, in: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), pp. 560–566.

[50] Viana, M., Alencar, P., Lucena, C., 2015. A Metamodel Approach to Developing Adaptive Normative Agents, in: 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), pp. 88–91.

[51] Warmer, J., 2003. The Object Constraint Language: Getting Your Models Ready for MDA. Addison-Wesley Professional.

[52] Wilensky, U., 1999. NetLogo. Center for connected learning and computer-based modeling, Northwestern University. Evanston, IL.

[53] Zhang, M., Verbraeck, A., 2014. A composable PRS-based agent meta-model for multi-agent simulation using the DEVS framework, in: Proceedings of the 2014 Symposium on Agent Directed Simulation, Society for Computer Simulation International, San Diego, CA, USA.

[54] Zilske, M., Nagel, K., 2017. Software Architecture for a Transparent and Versatile Traffic Simulation.

[55] Zilske, M., Schröder, S., Nagel, K., Liedtke, G., 2012. Adding freight traffic to MATSim. VSP Working Papers, TU Berlin. .