A Model-Driven Engineering Process for Agent-Based Traffic Simulations

Alberto Fernández-Isabel, Rubén Fuentes-Fernández
Universidad Complutense de Madrid, Madrid, Spain
afernandezisabel@ucm.es, ruben@fdi.ucm.es

Keywords: Traffic, Simulation, Agent-Based Modelling, Model-Driven Engineering, Modelling language, Metamodel, Code Generation.

Abstract: Traffic has an important impact in many aspects of our everyday life, from healthcare to transport regulation or urban planning. Given its complexity, the study in real settings is frequently limited, so researchers resort to simulations. However, realistic simulations are still complex systems. Its development frequently requires multidisciplinary groups, where misunderstandings are frequent, and there is a great variety of potential theories and platforms to consider. In order to reduce the impact of these issues, the Model-Driven Engineering (MDE) of simulations has been proposed. It is focused on developing mainly through models and their semi-automated transformation. Nevertheless, an effective approach of this kind requires the availability of infrastructures that include modelling languages, transformations, tools, and processes to use them. This work presents a MDE process for traffic simulations. It introduces a modelling language and makes uses of available infrastructures in its tasks. The process guides users in creating tailored models for their simulations, and transforming these to code. A case study that uses an existing model for drivers' behaviour and an already available platform to develop a simulation illustrates the approach.

1 INTRODUCTION

Road traffic is a complex phenomenon present in our everyday lives. Its analysis is relevant in studies of aspects as different as healthcare, road safety, transport regulation, fuel taxes, insurances, or urban planning (Elvik et al., 2009; Ward, 2004). Multiple factors make studying traffic on its actual settings hard and costly, for instance (Pursula, 1999): the involvement of living beings; the size of the areas to consider; the heterogeneity and number of elements participating in it; and the need of monitoring infrastructures. This situation has led researchers to make an extensive use of simulations to study it.

The simulation of complex systems presents its own challenges (Galán et al., 2009). It frequently requires integrating multiple theories, needed to deal with the different facets in the study and types of participants in the phenomenon. This demands multidisciplinary teams, where experts have different backgrounds. This heterogeneity produces misunderstandings, which in turn make difficult to guarantee the right alignment between theories, the setting of the study, and the final simulation, and to validate the results (Axtell & Epstein, 1994).

In order to address these issues, researchers have proposed using Model-Driven Engineering (MDE) to develop simulations (Fuentes-Fernández et al., 2012). MDE (France & Rumpe, 2007) is a paradigm for software development focused on *models*. These gather most of the required information (e.g. requirements, design, features of target platforms, or tests). Models must be compliant with explicitly defined Modelling Languages (MLs) to enable their semi-automated processing using *transformations*. Transformations are used, for instance, to merge models and generate code and documentation. Working with this infrastructure, the development of simulations becomes an iterative process of adding and modifying information in models and transformations. The main advantage is that experts make explicit all the information required to develop the simulation, and they can use for it different and tailored MLs in an integrated way. Moreover, they can reuse development artefacts more easily than in traditional approaches, as those are described at a higher level of abstraction than code.

The work presented in this paper follows this line. It provides a MDE process and its related infrastructure to develop traffic simulations.

The basis is an extensible Traffic ML (TML), suitable to integrate new concepts and theories. The language foundation lies on Agent-Based Modelling (ABM) (Axtell & Epstein, 1994) and the Driver-Vehicle Environment (DVE) model (Amditis et al., 2010). ABM has the *agent* as its main primitive. An agent is an entity situated in an environment. There, it interacts according to its capabilities with other agents and non-agent objects. The agent has a state that is frequently modelled in terms of mental entities, such as information and goals. The DVE model describes the main categories of elements participating in traffic (i.e. those in its acronym) and their structural relationships. It considers that traffic dynamics and the behaviour of its participants largely depends on their mutual influences. The proposed TML integrates these approaches, and adds extension mechanisms based on inheritance and composition relationships.

Traffic experts and *simulation designers* work in a process with two stages using the TML. The first one is focused on traffic issues, and is platform-independent. The second one is intended to develop the design and code of the simulation, and is platform-oriented.

Traffic experts start modelling with the TML the concepts of the specific theories to use in their simulation. For instance, the TML does not specify any actual model of decision-making for agents, so it must be added. If there are models of these theories available from previous projects, they can reuse them. Then, experts use these theoretical models to build the specifications of the simulation that represents the actual setting to study.

Simulation designers refine the concepts of the experts' models to design abstractions. The process suggests using as the target ML one of an Agent-Oriented Software Engineering (AOSE) methodology (Henderson-Sellers & Giorgini, 2005). In this way, the MLs of both experts and designers will be already partially aligned thanks to the use of agent abstractions, even if these do not have exactly the same semantics. After the mapping, designers use an existing general MDE methodology to make the transition to code. Many AOSE methodologies already apply MDE, so they are a suitable choice.

The infrastructure to work with these elements is based on several Eclipse projects for MDE (Eclipse, 2015). It includes a model editor for the TML. There are also transformations for different purposes. For instance, they implement the mappings between the traffic and designer MLs, as those of the case study.

The case study considers the reactive model for drivers' behaviour described by Ehlert & Rothkrantz (2001). This model proposes a reasoning cycle for people, and behaviour rules for different manoeuvres. The case study specifies it using the proposed TML, and uses this specification to define several elements for a simulation. Then, it applies the INGENIAS AOSE methodology (Pavón, Gómez-Sanz & Fuentes, 2005) to map the models to design concepts and generate the simulation code.

The rest of the paper is organised as follows. Section 2 introduces the main concepts of MDE. This is the foundation of the approach presented in Section 3, which includes the TML (see Section 3.1) and the process that uses it (see Section 3.2). The case study in Section 4 illustrates the application of the approach. Section 5 compares this with related work. Finally, Section 6 discusses the conclusions and future work.

2 MDE

MDE (France & Rumpe, 2007) is a general approach to software development organised around *models*. Engineers use their models to specify systems at different abstraction levels (i.e. from the experts' theories and requirements to the final design) and from different perspectives (e.g. those of final users, architecture, performance, or security). From them, semi-automated *transformations* generate other artefacts, e.g. code and tests. In order to make possible this processing, models must conform to well-defined MLs. This characteristic requires that MLs have explicit definitions to validate models against them.

There are different alternatives to define MLs. In the case of MDE, most of languages are graphical and graph-oriented, i.e. they depict graphs of concepts connected by links and with properties (Bézivin, 2006). The Unified Modelling Language (UML) (OMG, 2013) is a typical example of them. For these MLs, *metamodels* are the most popular means to define them.

A *metamodel* is a model itself that specifies the primitives and constraints of a ML. Metamodels are defined with meta-modelling languages, such as the Meta-Object Facility (MOF) used to specify UML, or Ecore in Eclipse projects. (Mu et al., 2010). The meta-modelling primitives of these languages allow defining types of graphs. For instance, in Ecore, an instance of *EClass* is a node, and thus a type of entity. An *EClass* instance groups *EAttribute* (i.e. properties of primitive *EDataTypes*) and *EReference* (i.e. binary and directed links between two *EClass* instances) instances. Besides these, Ecore includes,

among other concepts, an inheritance relationship (i.e. *ESuperType* reference) and the *EPackage* concept to group elements.

Regarding transformations, there are also multiple alternatives. Some works use specific languages to write them and engines for their execution. These languages range from completely procedural to fully declarative. Examples of this group are the Query-View-Transformation (QVT) standard of the OMG or the ATLAS Transformation Language (ATL) (Bézivin, 2006). Other works propose the use of general-purpose programming languages to write these transformations. An example of this case are the modules of the INGENIAS Development Kit (IDK) (Pavón, Gómez-Sanz & Fuentes, 2005). The first approach has the advantage of making clearer the relations between the source and target elements that transformations map. The second one makes use of mainstream expertise and tools, and facilitates fine-tuning the execution of transformations.

Engineers use the previous infrastructure in development processes. These are generally iterative and incremental. In each iteration, engineers refine their models by adding or modifying some information, e.g. considering new requirements or platform specific features. These tasks can involve introducing manually the new information, or running transformations to change and create artefacts. Both models and transformations are frequently reused from previous projects with similar needs. Examples of this kind of process are the Model Driven Architecture (MDA) (Kleppe, Warmer & Bast, 2003) for object-oriented developments, or PASSI and INGENIAS (Henderson-Sellers & Giorgini, 2005) for agent-oriented ones.

The applicability of these processes largely depends on the availability of support tools. For this reason, most of developments adopt Ecore instead of MOF to work with MLs, as there is an extensive support in the Eclipse MDE projects for the former. Researchers frequently extend the basic support provided by platforms and implement tailored tools for their processes. This is the case of INGENIAS with the IDK tool (Pavón, Gómez-Sanz & Fuentes, 2005). The IDK offers its own modelling environment. It supports defining both MLs and models compliant with them. It is also able to integrate modules to carry out transformations. In the case of model-to-text transformations, the IDK modules make use of templates. These are XML-like files with marked elements to indicate where to inject information from models.

3 DEVELOPMENT OF TRAFFIC SIMULATIONS

The MDE approach to develop traffic simulations is based on two main components: the TML to describe traffic settings (see Section 3.1), and a process that uses it (see Section 3.2).

3.1 TML

The TML is intended to describe the traffic settings to study. This requires considering two levels. The structural one indicates what elements are present and their features, and the behavioural one how they change over time. Traffic studies do not agree in common models for these aspects, given their variety of needs and theories. The ML acknowledges this situation and builds a flexible and extensible set of primitives, able to accommodate different conceptual frameworks. For this purpose, it adopts two widely used approaches as its basis, ABM (Axtell & Epstein, 1994) and the DVE model (Amditis et al., 2010). It organises their concepts in conceptual clusters using hierarchy and composition relationships. Fig. 1 shows the resulting metamodel described with Ecore (Steinberg et al., 2009).

The metamodel has two root elements. All the concepts inherit (not shown in Fig. 1) from the *GeneralElement EClass*. Instances of this element can be linked through arbitrary relations, represented as instances of the *GeneralRelationship EClass*. Being both root elements instances of *EClass*, their subtypes can have specific attributes. Most of concepts in the metamodel also have a *composes* relationship, which allows arbitrary whole-part hierarchies of constituents of the proper types.

Conceptually, the metamodel is organised around three clusters, that correspond to the description of the features and state of people (i.e. *features and internal state* cluster), the environment including the vehicles (i.e. *environment* cluster), and the decision-making of people (i.e. *interactions and decisions* cluster). These clusters pursue grouping concepts strongly connected in the DVE model and having clearly defined interaction points with other clusters.

The central concept of the TML is the *Person*. Its full description corresponds to the *features and internal state* and the *interactions and decisions* clusters. It represents any human participating in traffic, e.g. drivers and pedestrians from the DVE model. The *Person* is an intentional agent, as conceptualised in many ABM and AOSE (Henderson-Sellers & Giorgini, 2005) works. This means that it is modelled as pursuing certain *Goals*,

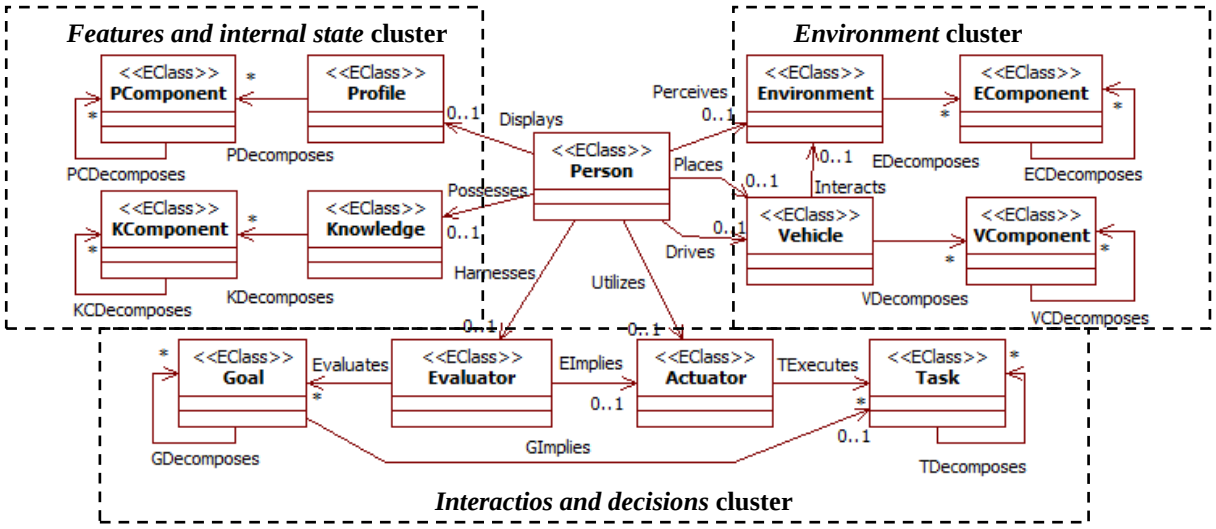

Figure 1: Excerpt of the traffic metamodel. Dotted rectangles indicate the main conceptual clusters. Inverse references are available for all the depicted associations.

that it can achieve through the execution of the *Task*s linked to them by the *GImplies* relationship. For instance, a driver has a goal of *driving safely* linked to a task of *checking distances with other vehicles*. An agent also has an internal state characterised in terms of its *Profile*, which represents its attributes, and *Knowledge*, to represent the mental entities it considers. Examples of parts of the profile are gender, age, or the type of driving according to past experiences; parts of knowledge can be information on driving norms, the streets in a city, or the best way to overtake another vehicle.

The *environment* cluster is used to describe the non-agent elements that can appear in a traffic setting. They include the *Vehicle* and *Environment* concepts. *Person* instances can interact with instances of these concepts. In the case of drivers and passengers, they perceive information from *Vehicle* instances, and the external environment also through them. They only act on vehicles. Although all the groups of persons can interact with the environment, their potential actions differ. For instance, a driver can brake a vehicle, and a pedestrian hit its bodywork.

The elements that merge the previous ones correspond to the perceive-reflect-act cycle of agents. They are part of the *interactions and decisions* cluster. *Evaluator* instances take as input information from the other clusters to generate new one. In particular, they are responsible of stablishing what goals are not satisfied at a given moment. *Actuator* instances execute some of the tasks related to unfulfilled goals.

Both *Evaluator* and *Actuator* instances interact with instances of other concepts of the metamodel, e.g. to gather information from *Environment* instances or to check *Knowledge* instances. As these elements of the *interactions and decisions* cluster model parts of a *Person*, their interactions are constrained to those instances accessible through their *Person* instance. These relationships of accessibility are represented with the different structural references from the *Person EClass* to others in Fig. 1. For instance, an *Evaluator* instance of a given *Person* instance only knows those *Knowledge* instances linked to their person through instances of the *Possesses EReference*.

The previous concepts are the basis to build the types in the models of specific theories. A simulation runs instances of these types. The metamodel includes the *GeneralIElement* and *GeneralIRelationship EClasses* to represent these *model instances*. In a model, the instances of these meta-classes are linked to instances of the *GeneralElement* and *GeneralRelationship EClasses* respectively, which are their *model types*. These links can include cardinality adornments to specify multiple instances. The attributes and references of the *model instances* are a subset of those of their *model types*.

3.2 Process

The MDE process for traffic simulations that uses the TML is depicted in Fig. 2. It has three stages, the specification of the theoretical framework (activities

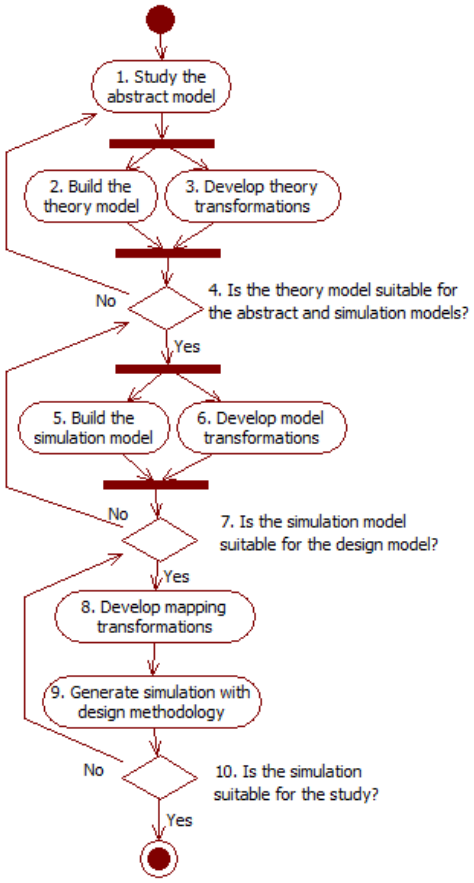

Figure 2: MDE process for traffic simulations.

1-3) and the simulation (activities 4-6), and the design of the simulation (activities 7-9).

The process starts analysing the original *abstract model* (i.e. the theory) in activity 1. The purpose is to identify the main concepts it manages.

Activities 2, *build the theory model*, and 3, *develop theory transformations*, ground the abstract model in a specific *theory model*. This is an instance of the TML that defines the main types of concepts and relationships of a theory (i.e. the *abstract model*). Researchers categorise the theory concepts as subtypes (i.e. instances) of those types in the TML. Relationship types must also be added at this step, as only very general ones are included in the TML. In this classification, researchers are guided by the description of the elements in the metamodel (see Section 3.1). Some information can be automatically added using specific transformations. For instance, a model can require that every instance of *Person* is related to at least one instance of *Goal* and another of *Task*, and these must be linked. Traffic experts and simulation designers jointly carry out these tasks.

In decision 4, traffic experts evaluate the theory model considering the abstract model it represents and the simulations to build. In particular, they need to consider whether it provides enough guidance and expressive power to specify those simulations. They can decide to refine it or to move to the simulation specification.

Activities 5 and 6 are similar to 2 and 3, but at the level of simulation specification. Traffic experts must create the *simulation model* that describes the simulation using as concepts those from the *theory model*. Specific transformations can appear here. For instance, experts can define transformations to initialise the simulation population (i.e. instances in the model) when this presents a regular distribution regarding the types of the simulation model.

The specification of the simulation ends with decision 7. If experts decide that the model is suitable for their study, they continue to the design stage; if not, they review the model.

Activity 8, *develop mapping transformations*, is the first of the design stage. Simulation designers are going to use in activity 9 an existing MDE methodology for the rest of the development. This has its own conceptual framework. Designers map the types in the simulation model to those of the design ML. This step may also need to consider the concepts implemented in the target simulation platform, as it could happen that not all the possible simulation-to-design mappings have the right semantics. These mappings are implemented with transformations. As the TML is agent-oriented, the use of AOSE methodologies in activity 9 facilitates these tasks.

The process ends with decision 10. Traffic experts and designers evaluate the generated simulation regarding the study to be carried out. They finish then the process or continue refining the development artefacts.

In the previous process, part of the developed artefacts are reusable between projects. The theory models and transformations are applicable whenever the same theoretical background is used. Mapping transformations are also highly reusable when the same theoretical background and design methodology are considered. Even simulation models can be partly reused when simulations have similar specifications.

Several support tools are available to help experts and designers. Model editors are available to specify the TML and the different models, and code editors for transformations. These are part of existing support in Eclipse MDE projects (Eclipse, 2015), although tailored tools can also be created.

4 CASE STUDY

The case study shows how to apply the MDE process (see Section 3.2) to develop the simulation of an existing study on traffic (Ehlert & Rothkrantz, 2001). It proposes a theoretical agent-based model for people, which is here the *abstract model*. There is an implementation of it as a test platform. Its development uses a traditional code-centric process with the Delphi object-oriented programming language. This analysis of the original work, with more depth, is the goal of activity 1.

The second step of the process is the specification of the *theory model* (see activity 2). This model grounds the main abstractions of the considered theoretical framework using the TML. In this case, the original work describes drivers and how they behave in the environment. Figs. 3-4 show this model. Associations without stereotypes are instances of *GeneralRelationship*. Their names are omitted when not used in the discussion.

The environment includes a *road* with *lanes*. These elements are *EComponent* instances connected by an *EDecomposes* relationship in the TML (see Section 3.1).

*Moving vehicle*s in traffic are *Vehicle* instances. In the studied work, a vehicle can provide information of its state. Two instances of *VComponent* model it. The *Vehicle state* instance has attributes to represents the intrinsic state of the vehicle, including speed or wheel angle. A vehicle also has an extrinsic state depending on its position on the road, which the *Vehicle position* instance represents. Determining this position requires accessing to information of the *Lane* instances. This access is represented with the new *moving in* instance of *GeneralRelationhip*.

A *driver* is characterised in terms of a profile with several behavioural parameters. These do not have a complex structure. Thus, this information is gathered in a *Behaviour parameters* instance of *Profile* as attributes. The person also perceives the external environment and vehicles, and generates *Knowledge* instances from this sensing. The *Vehicle distance* instance informs of the perceived distance between two vehicles, and the *Road information* instance of the distances to the *road* and *lane*s limits. In the theoretical model, decision-making needs to know the evolution of some data over time. In order to keep this record, the instances of *VComponent* and *KComponent* are considered as a subtype of a general instance *Timed data* of *GeneralElement*. This has a *time* attribute to indicate the moment of its observation.

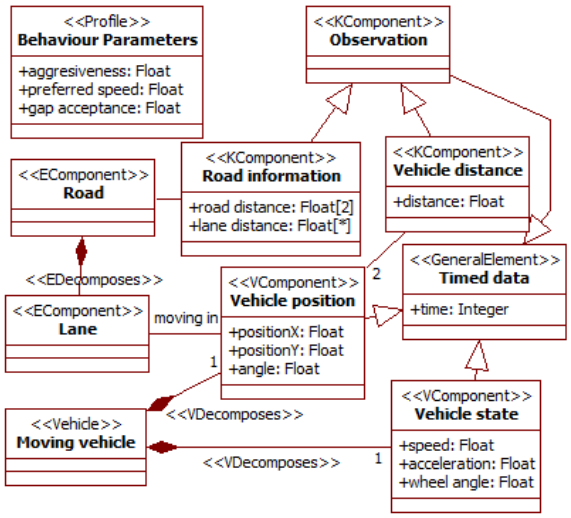

Figure 3: Theory model for the *features and internal state* and *environment* clusters.

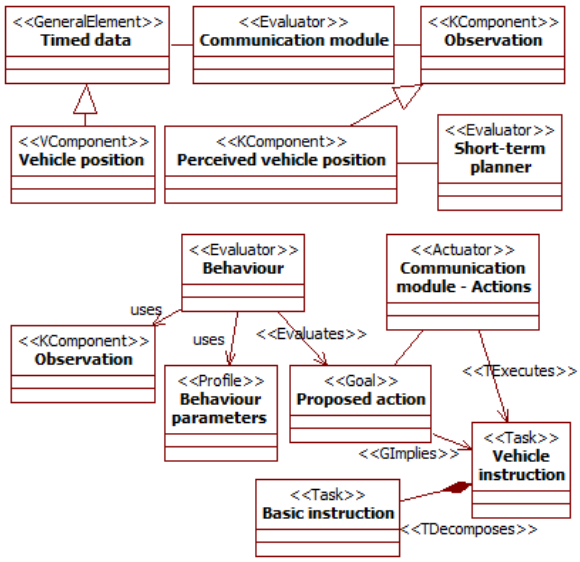

Figure 4: Theory model for the *interactions and decisions* cluster.

The rest of the abstract model is focused on how drivers behave. Being agents, they follow a perceive-reason-act cycle with several steps.

In a first step, they update their internal model of the world using perceived data. In the TML, this kind of modification is responsibility of evaluators. Following the original work, the theory model introduces an instance *Communication module* of *Evaluator* that takes as input *Timed data* and produces *Observation* instances of *KComponent* for the agents' internal state. For instance, this module is responsible for perceiving the *Vehicle position* in

the environment and generating its own instance *Perceived vehicle position* to represent it.

After processing sensor information, the agents update the expected future positions of other vehicles using a short-term planner. This corresponds to another instance of *Evaluator* that creates new instances of *Perceived vehicle position*.

Agents use the updated internal state to make decisions on the potential actions they can carry out. The abstract model refers to these decisions as *behaviours*. With the TML, these are modelled as *Behaviour* instances of *Evaluator*. They use available information to generate *Proposed action*s. that here play the role of *Goals*. In (Ehlert & Rothkrantz, 2001), drivers do not have explicit goals, but merely execute actions among those available. For this reason, the potential actions are seen in this theory model as goals.

The final step is the acting itself. The *communication module* selects among the *Proposed actions* those to perform actually, and sends to the vehicle the instructions. In the theory model, this corresponds to an instance *Communication module – actions* of *Actuator*, different from the previous instance *communication module* of *Evaluator*. It produces for the selected goals instances of *Vehicle instruction*, which are *Tasks*. These instances are decomposed into *Basic instructions*. These are in turn executed by instances of *Moving vehicle*. According to the TML, the actuator instance can only access to one *Moving vehicle* instance, which is that belonging to its *Person* instance.

The previous theory model does not impose a hard ordering of tasks: these can be executed as soon as their required information is available. In case that some tasks from the abstract model require strict ordering, the related TML tasks can produce specific instances of *KComponent*. Tasks starting sequences would produce these instances, that tasks following them would consume.

A *Driver* instance of *Person* links all the previous instances of *Profile*, *Knowledge*, *Evaluator*, and *Actuator*. It represents people participating in traffic according to the abstract model.

In this case, activity 3 does not create transformations that add information. Decision 4 considers this theory model suitable to continue with the following activities.

The simulation model can describe some sample settings of the original work. There are several types of driver that differ in their behaviour parameters. For instance, the “grandpa” and “young aggressive” drivers have different values for preferred speed or gap acceptance. Activity 5 models this with new subtypes of *Driver*. These subtypes only provide default values for some attributes of its base type. In a similar way, subtypes of *Road* and *Lane* can be introduced if needed to characterise different types of roads, e.g. cities and highways. Although here only attributes are considered, the TML also supports adding relationships.

As previously for the theory model, activity 6 does not create transformations. Decision 7 accepts this simulation model as input for the design stage.

The last specific activity of the process maps the previous concepts to those of the chosen design methodology. For this case study, this methodology is INGENIAS (Pavón, Gómez-Sanz & Fuentes, 2005), which is an AOSE and MDE methodology.

Thanks to the common background of the traffic and INGENIAS MLs in agent-based approaches, they share several similar concepts. The *Person*, *Task*, and *Goal* concepts from the TML are respectively *Person*, *Task*, and *Goal* concepts in INGENIAS. INGENIAS offers different concepts to represent metal entities of agents. In general, factual information as that represented by the TML *Knowledge* corresponds to INGENIAS *Fact* instances. The concepts mapped here have equivalent relationships in the traffic and INGENIAS MLs.

There are other concepts without that direct correspondence. INGENIAS does not distinguish between evaluators and actuators. It only considers tasks that process information. There are *mental state managers* and *mental state processors* that are somehow similar to evaluators and actuators respectively. However, they are intended to encapsulate specific algorithms, and not to describe the links between tasks, goals, and information. For these reasons, TML *Evaluator* and *Actuator* instances are mapped to INGENIAS *Task* instances.

In general, the INGENIAS ML has a richer model to represent information processing, decision making, and acting. These aspects are the core of its modelling. The TML has more primitives to represent the environment of agents, in particular for traffic.

The previous mappings can be implemented as transformations using available languages to transform between models serialised as XML. This is the case of the Eclipse-based tools for the TML and the IDK. These transformations allow generating from the simulation models the INGENIAS models, which are the design models of this case study. From them, a usual development with this methodology (see activity 10) can generate the code of the final simulation. In the case of

INGENIAS, the usual target platform is built over Jade (Bellifemine, Caire & Greenwood, 2007), which supports distributed agent-oriented systems. Nevertheless, the process is general, so other simulation platforms can be used, e.g. MATISSE (Zalila-Wenkstern, Steel & Leask, 2009) or MATSim (Waraich et al., 2015).

5 RELATED WORK

The kind of development for traffic simulations proposed in this work is related to several lines of research. It has to deal with conceptual frameworks and MLs for this domain, as well as development approaches for these simulations.

As said in the introduction, traffic studies have a wide variety of concerns and settings. There are works focused on, for instance (Elvik et al., 2009; Pursula, 1999; Ward, 2004), road congestion, safety, parking, or factors affecting driving. This variety makes useful different theories and simulation approaches depending on the actual study.

Conceptual frameworks are quite diverse (Barceló, 2010; Pursula, 1999). Nevertheless, some key concepts are recurrent in them: people (mainly drivers, but sometimes also pedestrians), vehicles, and the environment (most of times at least the road where vehicles move, but sometimes only the driver's visual field). Most of reviewed works fail to deal with this variety, as they fall in one of two extremes: they try to be too comprehensive (and become too general to provide guidance when modelling), or they consider a narrow domain (what drastically reduces its applicability outside it).

The type of considered simulation also affects the concepts available to model it. There are macroscopic simulations (e.g. with models for traffic flows based on gas and fluid dynamics (Helbing et al., 2001)), microscopic ones (e.g. with cellular automata (Maerivoet & De Moor, 2005)), and mesoscopic ones (e.g. combining discrete-event and queues approaches (Burghout, Koutsopoulos & Andreasson, 2006)). For these simulations, researchers use ad-hoc implementations for specific studies or platforms of general use (Barceló, 2010). In them, there are different abstractions to develop simulations. These depend on the underlying conceptual framework, but also on the code components.

Literature has proposed agent-based approaches to integrate these perspectives (Doniec et al., 2008), both at the level of conceptual frameworks and implementations. There are several frameworks considering the agent concept, such as MATSim (Waraich et al., 2015), MATISSE (Zalila-Wenkstern, Steel & Leask, 2009), or the work of Radecký and Gajdŏs (2006). Nevertheless, its actual definition presents a great variability. For instance, MATSim agents define their routes and times, but use algorithms and utility functions to optimise routes. In MATISSE, agents are closer to those of the presented approach, with interaction, planning, and task modules. Radecký and Gajdŏs base their implementation in Agent Behaviour Diagrams (ABD), a form of activity diagrams that can be implemented as finite state machines in the Jade platform (Bellifemine, Caire & Greenwood, 2007). Such platforms are tied to the concepts they provide for simulation, as it happens with non agent-oriented platforms. Experts need to adhere to their conceptual frameworks for their simulations, independently on whether they are or not the best suited for their studies. For instance, traffic works considering statistical distributions of traffic, do not need the overhead of agent models as those in these platforms.

In this context, the development process can play a key role to adapt the existing infrastructures to the actual needs of studies. The can provide guidance on how to perform certain development tasks. Though there are few reports on these issues, available information mainly corresponds to traditional code-centric approaches (Bellifemine, Caire & Greenwood, 2007; Ehlert & Rothkrantz, 2001; Radecký and Gajdŏs, 2006). These give guidelines to structure code and the related algorithms. When they use models (Bellifemine, Caire & Greenwood, 2007; Radecký and Gajdŏs, 2006), their purpose is enabling design, discussion, and documentation. Programmers manually code them. There are few examples of MDE approaches, like (de Lara, Vangheluwe & Mosterman, 2006). They are illustrative of some advantages already mentioned for our approach, such as the high-level specification of simulations and making explicit all the relevant information through models. However, these examples frequently fail to involve some groups of experts. They try to cover the complete development cycle, from requirements to coding, while their infrastructures are typically oriented to only some stages. Moreover, they use formalisms and languages that are not widely used in the community of MDE. The work in (de Lara, Vangheluwe & Mosterman, 2006), which bases development on graph rewriting techniques, is an example of it.

6 CONCLUSIONS

This paper has introduced an approach for the MDE of traffic simulations. It proposes a complete process with two stages, a first one focused on traffic experts and a second one on simulation designers. The first stage uses a specific TML and standard tools from Eclipse projects. This makes possible providing tailored tools for these experts. The second part links to existing MDE AOSE methodologies for the low-level (i.e. platform-oriented) design. The result is a process that provides new infrastructures to support traffic experts, while it takes advantage of existing efforts to support the specific tasks of simulation designers.

Regarding the TML that is the core of the work, its design balances providing modelling guidance and flexibility. The first feature is achieved adopting as its conceptual basis two widely used approaches in the area, the DVE model (Amditis et al., 2010) and ABM (Axtell & Epstein, 1994). The flexibility is achieved incorporating mechanisms to easily extend the language through inheritance and composition, both at the level of types and instances in models. These combined features allow keeping the TML set of primitives intentionally simple, while supporting its adaptation to the specific needs of studies.

The case study has illustrated the features of the process by modelling an existing traffic model (Ehlert & Rothkrantz, 2001), and using and AOSE methodology (Pavón, Gómez-Sanz & Fuentes, 2005) for design and coding. The development of the simulation mainly makes use of model editors. It uses transformations in the transition from the TML specifications to INGENIAS models. At that point, a standard AOSE development with that methodology takes place.

The work presented here still has several open issues. The first one is considering additional theories and frameworks in order to check the validity of the TML, and incorporating new primitives if needed. Secondly, though the case study needed few transformations, this is not always the case. Besides adding information to models, they can also be useful to check automatically properties. The use of transformation languages (Bézivin, 2006) helps non-experts in MDE to understand mappings, but developing transformations is still a hard work. The exploration of transformation-by-example approaches (García-Magariño, Gómez-Sanz & Fuentes-Fernández, 2009) to generate automatically transformations from model instances is an alternative. Finally, though design methodologies (e.g. INGENIAS in the case study) can consider distribution issues for simulation, their figures regarding components are not usually those of complex traffic simulations. Further experimentation is required here to identify potential requirements in design for these aspects.

ACKNOWLEDGEMENTS

This work has been done in the context of the projects “Social Ambient Assisting Living - Methods (SociAAL)” (TIN2011-28335-C02-01) supported by the Spanish Ministry for Economy and Competitiveness. Also, we acknowledge support from the “Programa de Creación y Consolidación de Grupos de Investigación” (UCM-BSCH GR35/10-A).

REFERENCES

Amditis, A., Pagle, K., Joshi, S., Bekiaris, E., 2010. Driver-Vehicle-Environment monitoring for on-board driver support systems: Lessons learned from design and implementation. *Applied Ergonomics*. 41(2), p. 225-235.

Axtell, R., Epstein, J., 1994. Agent-based modeling: understanding our creations. *The Bulletin of the Santa Fe Institute*. 9, p. 28-32.

Barceló, J. (ed.), 2010. *Fundamentals of traffic simulation*. International Series in Operations Research & Management Science, 145, Springer. Heidelberg, Germany.

Bellifemine, F. L., Caire, G., Greenwood, D., 2007. *Developing multi-agent systems with JADE*, John Wiley & Sons. Hoboken, NJ, USA.

Bézivin, J., 2006. Model driven engineering: An emerging technical space. In: Lämmel, R., Saraiva, J., Visser, J. (eds.) *Generative and Transformational Techniques in Software Engineering*, Lecture Notes in Computer Science 4143, p. 36-64. Heidelberg, Germany: Springer.

Burghout, W., Koutsopoulos, H. N., Andreasson, I., 2006. A discrete-event mesoscopic traffic simulation model for hybrid traffic simulation. In *2006 IEEE Intelligent Transportation Systems Conference (ITSC 2006)*. Toronto, Canada, 17-20 September 2006. Washington, DC, USA: IEEE Computer Society. p. 1102-1107.

Doniec, A., Mandiau, R., Piechowiak, S., Espié, S., 2008. A behavioral multi-agent model for road traffic simulation. *Engineering Applications of Artificial Intelligence*. 21(8), p. 1443-1454.

Eclipse, 2015. Eclipse Modeling. [Online] Available from: http://eclipse.org/modeling/ [Accessed: 10 March 2015]

Ehlert, P. A. M., Rothkrantz, L. J. M., 2001. Microscopic traffic simulation with reactive driving agents. In *2001 International Conference on Intelligent Transportation Systems (ITSC 2001)*. Oakland, CA, USA, 25-29 August 2001. Washington, DC, USA: IEEE Computer Society. p. 860-865.

Elvik, R., Vaa, T., Erke, A., Sorensen, M., 2009. *The handbook of road safety measures*, Emerald Group Publishing. Bingley, UK, 2nd edition.

France, R., Rumpe, B., 2007. Model-driven development of complex software: A research roadmap. In *2007 Future of Software Engineering (FoSE 2007)*. Minneapolis, MN, USA, 23-25 May 2008. Washington, DC, USA: IEEE Computer Society. p. 37-54.

Fuentes-Fernández, R., Hassan, S., Pavón, J., Galán, J. M., López-Paredes, A., 2012. Metamodels for role-driven agent-based modelling. *Computational and Mathematical Organization Theory*. 18(1), p. 91-112.

Galán, J., Izquierdo, L., Izquierdo, S., Santos, J., del Olmo, R., López-Paredes, A., Edmonds, B., 2009. Errors and artefacts in agent-based modelling. *Journal of Artificial Societies and Social Simulation*. 12(1), p. 1.

García-Magariño, I., Gómez-Sanz, J. J., Fuentes-Fernández, R., 2009. Model transformation by-example: an algorithm for generating many-to-many transformation rules in several model transformation languages. In: Paige, R. F. (ed.) *Theory and Practice of Model Transformations*, Lecture Notes in Computer Science 5563, p. 52-66. Heidelberg, Germany: Springer.

Helbing, D., Hennecke, A., Shvetsov, V., Treiber, M., 2001. MASTER: macroscopic traffic simulation based on a gas-kinetic, non-local traffic model. *Transportation Research Part B: Methodological*. 35(2), p. 183-211.

Henderson-Sellers, B., Giorgini, P. (eds.), 2005. *Agent-oriented methodologies*, IGI Global. Hershey, PA, USA.

Kleppe, A. G., Warmer, J., Bast, B., 2003. MDA Explained: *The Model Driven Architecture - Practice and Promise*, Addison-Wesley. Reading, MA, USA.

Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L., 2012. Recent development and applications of SUMO - simulation of urban mobility. *International Journal on Advances in Systems and Measurements*. 5(3-4), p. 128-138.

de Lara, J., Vangheluwe, H., Mosterman, P. J., 2006. Modelling and analysis of traffic networks based on graph transformation. In *5th Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems (FORMS/FORMATS 2004)*. Braunschweig, Germany, 2-3 December 2004. Braunschweig, Germany: Technical University of Braunschweig, Institute for Traffic Safety and Automation Engineering. p. 120-127.

Maerivoet, S., De Moor, B., 2005. Cellular automata models of road traffic. *Physics Reports*. 419(1), p. 1-64.

Mu, L., Gjøsæter, T., Prinz, A., Tveit, M. S., 2010. Specification of modelling languages in a flexible meta-model architecture. In *4th European Conference on Software Architecture (ECSA 2010)*, companion volume. Copenhagen, Denmark, 23-26 August 2010. New York, NY, USA: ACM. p. 302-308.

Object Management Group (OMG), 2013. *OMG Unified Modeling Language (OMG UML), Version 2.5*. [Online] Available from: http://www.omg.org/ [Accessed: on 10 March 2015).

Pavón, J., Gómez-Sanz, J. J. Fuentes, R. (2005) The INGENIAS methodology and tools. In: Henderson-Sellers, B., Giorgini, P. (eds.) *Agent-oriented methodologies*, p 236-276. Hershey, PA, USA: IGI Global.

Pursula, M., 1999. Simulation of traffic systems - An overview. *Journal of Geographic Information and Decision Analysis*. 3(1), p. 1-8.

Radecký, M., Gajdŏs, P., 2006. Intelligent agents for traffic simulation. In *2008 Spring Simulation Multiconference (SpringSim 2008)*. Ottawa, Canada, 14-17 April 2008. San Diego, DC, USA: Society for Computer Simulation International. p. 109-115.

Steinberg, D., Budinsky, F., Paternostro, M., Merks, E., 2009. *EMF: Eclipse Modeling Framework*, Pearson Education. Upper Saddle River, NJ, USA, 2nd edition.

Waraich, R. A., Charypar, D., Balmer, M., Axhausen, K. W., 2015. *Performance Improvements for Large-Scale Traffic Simulation in MATSim*, Springer. Heidelberg, Germany.

Ward, S., 2004. *Planning and urban change*, SAGE. London, UK, 2nd edition.

Zalila-Wenkstern, R., Steel, T., Leask, G., 2009. A self-organizing architecture for traffic management. In: Weyns, D., Malek, S., de Lemos, R., Andersson, J. (eds.) *Self-Organizing Architectures*, LNCS 6090, p. 230-250. Heidelberg, Germany: Springer.