Proceedings of the 2019 Winter Simulation Conference
N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, eds.

COUPLED TRAFFIC SIMULATION BY DETACHED TRANSLATION FEDERATES: AN HLA-BASED APPROACH

Moritz Gütlein
Anatoli Djanatliev

Computer Networks and Communication Systems
Friedrich-Alexander University Erlangen-Nürnberg (FAU)
Martensstr. 3
Erlangen, 91058, GERMANY

ABSTRACT

Typically, there is more than one possible answer to the question of how to model and simulate a problem. Combinations of different simulation paradigms and techniques are currently in vogue and potentially necessary to meet certain requirements. Besides clear advantages, model combinations tend to increase the complexity of the overall model building process. Even if different paradigms are coupled via a middleware, another use case might require reimplementing the connections. Additionally, users need increased effort to understand how the results have been generated in composed models. If it is not clear how data has been transformed from input to output, the trust in simulation is impaired. In order to counteract, a concept for reusable translation between submodels is proposed, leading to a better transparency regarding the data flows. We applied the High Level Architecture based approach to connect a macroscopic, a mesoscopic, and a microscopic traffic simulation.

1 INTRODUCTION

Using modeling and simulation to answer complex real-world questions can result in the urge to choose between different ways to model a problem. The different options may vary in the required input data, modeling paradigms such as Agent-Based Simulation (ABS), System Dynamics (SD), or Discrete Event Simulation (DES), and key factors such as resolution level in terms of discrete step lengths, or precision of calculations. Decisions taken on how to model a problem affect the results significantly. In general, this can be described as a trade-off between computational performance and simulation accuracy. Moreover, not every question requires the simulation to be performed at the highest possible level of detail.

Sophisticated questions may typically not be answered by a single approach. Such questions might require the combination of different paradigms such as DES and SD, resulting in a Hybrid Simulation (Martinez-Moyano and Macal 2016). Eldabi et al. (2018) define Hybrid Simulation as the combination of at least two models out of DES, SD, or ABS paradigms. Likewise, in the field of traffic simulation, there is a need for multi-level and hybrid simulations as previously shown by Gütlein et al. (2018). Major reasons named are performance, the lack of data availability, and the feasibility of large-scale simulations. If a city-wide traffic simulation is required, it can usually lead to problems regarding memory limits and the computational time to run the simulation. The problem gets even more notable when the simulation has to run faster than real time, in order to be able to predict upcoming traffic jams, or evaluate different strategies enhancing the current traffic situation (e.g. part-time shoulder use, dynamic speed limits, or adaptive re-routing suggestions). At the same time, the input data problem can be exposed. Imagine that there is the need for a microscopic traffic simulation applying an agent-based model (e.g. to evaluate traffic light programs). However, the usually available input data consists only of average speeds per road segments.

This data is not sufficient for the derivation of agents to form road traffic without further processing. On the other hand, a macroscopic traffic simulation that handles aggregated measures would be easily set up with the existing data. A coupled approach, in which a macroscopic simulation *feeds* a microscopic simulation with adequate information, may solve the mismatch of available data and desired model.

Besides, we face the problem that simulation might be seen in a crisis of confidence. Simulation results may not be considered trustworthy or meaningful (Uhrmacher et al. 2016). Confidence could be built if publicly available simulation tools were used and comprehensible models and parameters were published, which enables the reproduction of results. If model parts were well understood and tools themselves were reusable, how to improve further? In this work, we focus on the issue of the information transition between different submodels. Logic that translates incoming data might be distributed all over the simulators' and federates' code parts. The approach enhances reusability and creates a more transparent data flow that spreads from the data input to the resulting output. Therefore, all logic that moves information from one submodel to another is encapsulated into translation units and detached from the submodels' logic. Tools and translators are connected using the High Level Architecture (HLA) as a middleware. The approach is illustrated by an example from the field of traffic simulation. We briefly explain the HLA and traffic simulation, introduce the concept and advantages of detached translation units, and depict how the concepts of HLA and detached translators are matching. We describe a traffic simulation consisting of a macroscopic, a mesoscopic, and a microscopic simulator connected through exemplary implementations of the presented translation units over HLA. Finally, we evaluate the overhead of detaching translators into own federates compared with a classic integrated version, discuss limitations, and give an outlook for future research.

2 RELATED WORK

In order to grasp the motivation and the suggested approach, we give information about HLA, multi-level traffic simulation, simulation credibility, reusability, and information flow between submodels. HLA is a standard for distributed simulation. Standardized as IEEE 1516, it offers a set of services that allow distributed simulations. A single participating tool is called federate, while the set of federates that form one common simulation is called federation (Dahmann et al. 1998). Information exchange and time synchronization between federates is handled by the Runtime-Infrastructure (RTI). The Federate Object Model (FOM) defines all possible interactions and objects including their attributes that can be exchanged via the simulation bus. There are multiple RTI implementations available, both commercial and open source. HLA has been used for years in the field of coupled traffic simulations (Ozaki et al. 2000).

Traffic simulation is classically divided into four categories: macroscopic, mesoscopic, microscopic, and submicroscopic. In macroscopic simulations, traffic models deal with average speeds, link capacities, and link volumes. Traffic can be seen as a flow and there is no representation of individual traffic objects. Individual representations are available in mesoscopic models, although it is possible to have some kind of prototype that is an aggregate of multiple entities (Tolujew and Alcalá 2004). Microscopic traffic simulation has representations for each road user having a separate behavior model. Models expecting a more detailed representation of objects can be built by the submicroscopic approach, e.g. the behavior of vehicles' emergency brake systems (Gütlein et al. 2018). Burghout et al. (2005) state that microscopic traffic models allow the evaluation of intelligent traffic systems, because of their level of detail. However, they admit that this level of detail uncovers new problems such as calibration and input preparation. Consequently, it is more error prone than coarser models. Using a hybrid traffic simulation allows them to combine the advantages of the different models. The motive of Sewall et al. (2011) for the use of a hybrid approach lies in the efficient simulation of large-scale scenarios. They use Poisson processes to transfer traffic information between a continuous and a discrete regime.

Pawlikowski et al. (2002) complain that many simulation studies in the field of telecommunication networks cannot be trusted. They introduce the term *credibility crisis*. In order to identify mistakes, knowledge about the experiments is necessary. Dalle (2012) criticizes that a lack of published information regarding a simulation experiment hinders the reproducibility of results. This can be caused by hidden parameters (e.g. inside a commercial tool), which are not even accessible to the authors, or information that is not released although available. Accordingly, Yilmaz et al. (2014) mention that the credibility gap in simulation results is growing and therefore reproducibility of studies is urgently needed. A further problem exists if the origin of results cannot be reproduced, which leads again to problems in terms of assessing and reproducing the results. Ruscheinski and Uhrmacher (2017) address this problem of provenance of simulation results. They see one of the main issues for this in the question of how a model was created and propose a provenance model that allows reconstructing the relations and flows that lead to a simulation study. Taylor et al. (2018) discuss the importance of reproducibility and add that there are critical fields (e.g. military or medicine), where data or techniques cannot be released.

In addition to the credibility benefits, component reuse can save a lot of money and time. This applies to software development in general and to any other building process. The ideas of object-oriented programming (e.g. encapsulation) can support this concept (Meyer 1987). Balci et al. (2017) see model reuse in simulation as a difficult topic in general, as a model is usually created for a specific purpose. However, they mention that some communities (e.g. micro-electronics design) are successfully implementing model reuse. From our point of view, this applies also to the field of traffic simulation. HLA, in particular, is well known for interoperability, model reuse, and composability of submodels (Zhu et al. 2019). Related publications address design principles to empower reuse of federates (Kang et al. 2018) or propose frameworks that support the development of HLA-Federates (Cox 1998). Klein et al. (1998) classify HLA as most promising technology for reusable (traffic) simulation. Although there is research done in the field of model and component reuse (Saulnier and Bortscheller 1994; Pos et al. 1996; Zhu et al. 2014), the topic of reusing the data conversion logic between models is quite disregarded.

Sung and Kim (2011) focus on the information flow between models and utilizes a converter to translate between a continuous and a discrete model. However, the authors do not address the reusability of the conversion logic for other tools. Radeski et al. (2002) were inspired by the Common Object Request Broker Architecture Component Model. Besides other points regarding the improvement of component reuse in HLA simulations, the authors are aware of data conversion problems (e.g. conversion of a double to long). They propose an additional layer that handles this custom data conversion in an interceptor manner, but inside a single simulation component. Oses et al. (2004) name different problems regarding component-based discrete simulation. Following the object-orientated-principles, they mention that mediators may translate between two specific components, but do not line out further details or directions.

To sum it up, there are publications regarding simulation model reuse and the information transfer between submodels. The disclosure and reuse of the model translation has not been considered so far. Although tools usually offer features for reproducibility and reusability, we face therefore the situation that a coupled model loses reproducibility after combining tools. The conceptual connection between different models might be anything but distinct and represents the critical point for reusing, reproducing, or adapting simulation models. This is where this paper steps in, it focuses on enhancing and reusing information transition between submodels implemented by different simulation paradigms and data models. To the best of our knowledge, we are the first to particularly address the reuse of the translation between submodels.

3 TRANSLATION UNITS

In general, differences between simulators are implying distinct data models (e.g. vehicles with positions vs. roads with volumes) and different model interfaces. A translation unit (TU) handles the bidirectional translation between two particular model interfaces. Hence, it translates a data tuple that matches a submodel to a tuple that matches another submodel and back. Therefore, a TU contains two unidirectional translators.

3.1 Data Dimension and Data Layer

In order to structure interfaces for the existing data models in simulation tools, information is partitioned into data dimensions and further into data layers. A dimension classifies the domain the information belongs.

In the case of traffic simulation, the dimension is *traffic*. If one would like to do a cross-domain simulation, e.g. simulate vehicular communication, *communication* would be a second dimension besides *traffic*. A data dimension consists of layers that belong exclusively to the dimension. Each layer is composed of a well-defined set of parameters including a **key**. Parameters have an identifier, a data type, and optionally a unit. Sticking to the traffic simulation example, we will depict this by introducing three data layers in the traffic data dimension: Macro, meso, and micro. **Road** and volume are the parameters of the macro layer. The meso layer consists of road, **vehicle (id)**, and route. Finally, the micro layer consists of road, **vehicle (id)**, route, lane, speed. All data tuples have a simulation time stamp. Besides simulation scenario information, these data definitions need to be accessible for researchers to empower transparency, understand relations between the submodels, and be able to reproduce simulation results and reuse components. The layering concept allows simulators assigned to the same layer to be seamlessly exchanged (e.g. use a microscopic traffic simulator from another organization).

3.2 Definition of Translation Units

A TU is a bidirectional converter function that combines two translators: An aggregator and a disaggregator translator. A TU handles information exchange between two different models, where both models have to be categorized to exactly one data layer per data dimension. A translator T is made up of an input layer L_I, an output layer L_O, and a conversion function f. f translates a tuple P_I from L_I's parameter set into a tuple P_O matching the definitions of L_O:

$$T = (L_I, L_O, f), \;\; f : L_I \to L_O, \;\; P_I \mapsto P_O$$

A TU is defined by two translators, TA (aggregation) and TD (disaggregation), and two layers of the same dimension L_A and L_D. The input layer of one translator (T_{L_I}) corresponds to the output layer of the other translator (T_{L_O}) and vice versa. Implicitly, the dimensions of the two translators match:

$$TU = (TD, TA, L_A, L_D), \;\; TD_{L_I} = TA_{L_O}, \;\; TD_{L_O} = TA_{L_I}, \;\; TD_{L_{I_D}} = TD_{L_{O_D}}$$

The purpose of explicitly defining a TU is to be able to detach the TU from the simulation wrappers. The schema of a traditionally coupled simulation over HLA can be seen in Figure 1. There may be some interface definition called C, which is defined in the FOM (see section 2). Both simulators are wrapped next to federates that publish and subscribe to an interaction or an object that can be directly represented by C. The simulation tools themselves may have completely different interfaces (e.g. A and B), which leads to the need of translating the information flow between simulator and federate (ambassador). Translation might be done by many pieces of code, entangled between HLA logic and communication with the simulator's API, compiled to one all-embracing executable. As a result, the translation is indistinct and thus, is breaking reusability and reproducibility of the whole simulation model in this step.

The objective of detaching the translation process into a separate module is to enable the use of the connected submodels' native interface definitions. This should encourage consciousness about the information transfer and conversion. The new schema is depicted in Figure 2. Consequently, no intermediate interface definition like *Interface C* is required anymore. This achieves modularity, which allows for component reuse, adaptive replacement of TUs during simulation runs, and exchange of TUs with other researchers. A translation federate skeleton handles all stuff apart from the translation itself. This facilitates the application by solely having to put in the translation algorithms. In addition to the executable, the translation model can easily be disclosed. This could be done by source, pseudo code, mathematical formulations, or all together. Because the TU skeleton is fixed and can be reused, only the sharing of code that really does the math is necessary. Even with no background in computer science, it should be possible to reproduce the steps, or look into the mathematic formulation of the translator. Other researchers may be encouraged to get in touch with the code snippets, challenge, and reuse them. This should improve trust in simulation results. The key point of this effort is that we assume the used tools and their models to be understood, reproducible, reusable, and available - in contrast to the current connection of different tools.

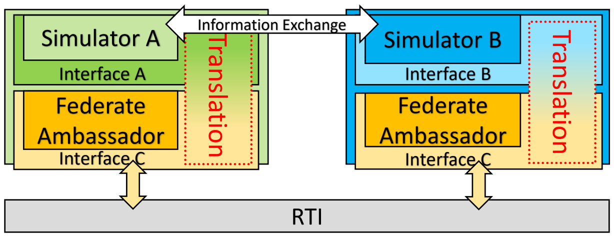

Figure 1: Coupling with embedded translation.

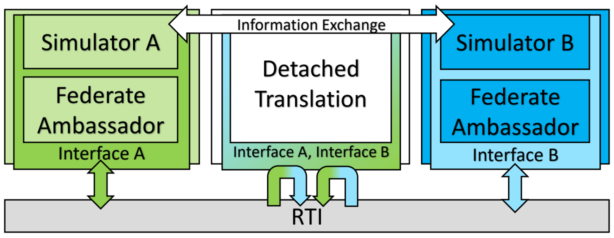

Figure 2: Coupling with detached translation.

3.3 Translation Unit Components

As briefly stated, the two main components of a TU are the aggregator component TA and the disaggregator component TD. This implies the hierarchical approach that data layers within one dimension can always be ordinarily ordered regarding their fidelity. The conversion function f needs mapping rules for each parameter in the target layer's parameter set. Parameters from the source layer's parameter set can be left out. As a consequence, f needs to be a surjective mapping of functions f_i (Figure 3).

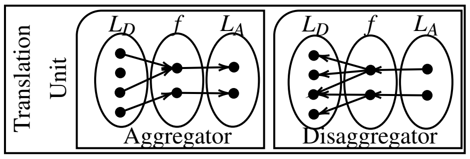

Figure 3: Translation Unit consisting of aggregator and disaggregator.

3.3.1 Aggregator

According to that claim, the aggregator handles the translation of data flowing from the disaggregation layer L_D to the aggregation layer L_A. In the simplest case, the translation is done by aggregating data from the more precise data model by calculating an average, e.g. the average speed $L_A^{\bar{S}}(r,t)$ for time t driven on a road r by using an average over the speeds L_D^S of all vehicles v that are driving on r at time t. Hence, we use the notion $L_A^{\bar{S}}(r,t)$, when a parameter $\bar{S}$ from a data tuple of layer A with key r and time t is accessed.

3.3.2 Disaggregator

In the opposite direction, the disaggregator component handles the translation of information flowing from the aggregation layer L_A to the disaggregation layer L_D. Unlike the aggregation process, information does not have to be reduced. Instead, additional information can be generated based on current state, historic data, and stochastic processes. An example from the traffic domain is the sampling of individual car arrivals out of an aggregated measure such as the traffic volume on a road segment for a given duration. If the arrivals of single vehicles on a road are expected to be independent of each other, a Poisson arrival process (PAP) can be used to translate a road's traffic volume V_R per time span Δt into discrete vehicle arrivals. Thus, additional information is generated.

3.4 Translation Unit Skeleton

The architecture allows proposing and implementing a general TU skeleton. It is fully functional and connected to the HLA federation. It provides interfaces for the two defined data layers and stubs for the two unidirectional translators. For every used TU, the skeleton can be reused and extended with the specific aggregator and disaggregator translation algorithm. As the skeleton does not need to be published by the authors of a study, redundancy in published and exchanged code is minimal. The translator stubs provide three functions: `receive(`L_I`)` to process incoming information, `send(`L_O`)` to send out information, and `loop()` to sample continuous processes. Additional helper functions can be added.

3.5 HLA and Translation Units

Data based partitioning allows the categorization of simulation models based on their data model and modeling paradigm. According to that, the classification of simulation tools are derivable and thus their natural interface definition ID as well. It is required that each used simulation tool is assigned to at most one data layer per dimension and at least one data layer in total. It is possible that different tools are assigned to the same data layer. Each tool must be wrapped next to an HLA federate that allows communication and synchronization between tool and federation. The federate wrapper should then provide the interface definition ID that belongs to its simulation tool. This enables the unmodified transfer of simulation related information between the simulation tool itself and another tool from the same layer or a corresponding TU.

A new simulation experiment can be started by defining the scenario's topology. The topology contains all used tools, their interfaces, spatial, temporal or modal responsibilities, and connections between tools. Subsequently, the HLA FOM is generated out of this information. This allows that simulators of the same layer can be exchanged seamlessly, without having to change any other parts. In addition, translators can be smoothly replaced without breaking the setup. Components can be externalized and connected remotely, which is useful when cooperating with companies that refuse to release algorithms or binaries.

4 APPLICATION ON HYBRID TRAFFIC SIMULATION

In this section, the presented approach will be clarified by applying it on a multi-level traffic simulation model. The overall simulation model combines a macroscopic, a mesoscopic, and a microscopic traffic model. For this purpose, the data model presented in subsection 3.1 will be used. As a first step, we are limiting the application on a circular road. We use a topology, where each tool is responsible for a single region. Neighboring regions are connected via border links (BL) that are visible to both responsible tools as shown in Figure 4. Tools are informed and aware of their responsible area and outgoing border links to other areas. A single simulator does not have any information about the number and type of its neighbors. Outgoing information (e.g. a vehicle enters the border link or the link's volume representation changes) is published according to the sending tool's native data model. More information about this data exchange concept can be found in Gütlein et al. (2018). We extended the concept slightly by integrating the translation units. The basic idea that a tool sends out information in its own data format still exists, while the part of receiving and interpreting unfamiliar data models is enhanced. With the new approach, traffic information that crosses borders is translated by the TU and thus delivered to the recipient in its corresponding data format. The C++ implementations of the translators are available on our GitHub repository (Gütlein 2019).

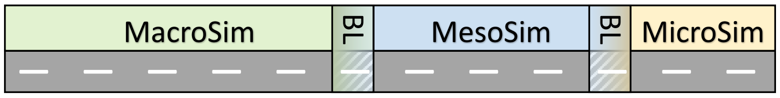

Figure 4: Traffic simulation topology with responsibilities and border links.

4.1 Translation Units

The macro layer deals with a parameter set containing a road and its current status (i.e., volume). In the micro (vehicle, road, route, lane, speed) and meso layer (vehicle, road, route), the parameter sets consist of vehicle and time-related attributes, where *road* is now an attribute itself describing a vehicle's position. According to the definition in subsection 3.2, two TUs TU_1 and TU_2 are declared:

$$\begin{aligned}&Dimension = \{traffic\},\ TU_1 = (A, B, macro, meso),\ TU_2 = (C, D, meso, micro)\\&Layers = \{macro = \{\{road, volume\}, traffic\},\ meso = \{\{road, vehicle, route\},\\&traffic\},\ micro = \{\{road, vehicle, route, lane, speed\}, traffic\}\}\end{aligned}$$

4.2 TU 1: Macroscopic and Mesoscopic

The transition between macro and meso is at the same time the transition between continuous time and discrete event simulation. Parameters that describe a system from the top have to be mapped to parameters that describe a system from inside. Therefore, the state of individual traffic participants need to be inferred from macroscopic computations and vice versa.

4.2.1 TU 1 A (Disaggregator): From Macro to Meso

One main question has to be answered: At what times do vehicles have to be spawned on a road to meet a certain set of macroscopic parameters? Furthermore, the behavior or properties of a generated vehicle must be clarified. That may be solely the vehicle's type in the mesoscopic case, or be broken down to the chosen lane, destination and route, the driver model, the preferred speed, or even the dimensions of the car for the microscopic case. Modeling the arrival of events in a discrete system can be done with PAPs. This requires the assumption that arrivals are independent of each other and the inter-arrival times between the events are exponentially distributed. The modeling of arrivals of incoming phone calls into an operator's network is a well-known application for this (Ibrahim et al. 2012). With Equation 1, the probability that $N(t)$ arrivals occurred during duration t can be calculated (Figure 5). λ describes the arrival rate. Assume that the incoming volume of a road is 2 cars per minute, λ would also be 2 per minute. Since we only know the arrival rate, we are interested in sampling when the next arrival event is likely. Thus, the next inter-arrival time is estimated. As mentioned, exponentially distributed inter-arrival times are assumed. Therefore, the cumulative distribution function (CDF) is used for this exponential distribution. The CDF (Figure 6) allows inferring probabilities for durations between two arrivals.

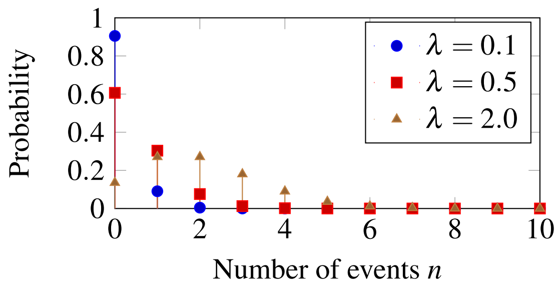

Figure 5: Poisson process.

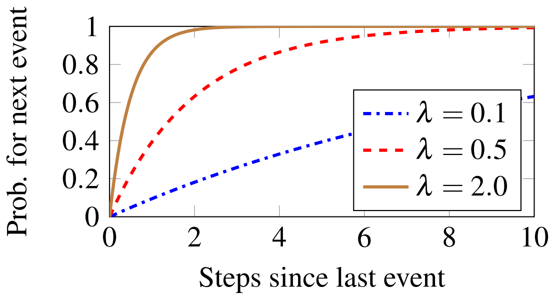

Figure 6: CDF.

Finally, the inverse of the CDF is used, which is resulting in Equation 2. To realize the translation, a random number $r \in (0, 1)$ will be generated from a uniform distribution before each arrival. Having the inverse CDF calculated, r is mapped to the next inter-arrival time i as depicted in Figure 7.

$$P\{N(t) = n\} = \frac{\lambda t^n}{n!} e^{-\lambda t} \tag{1}$$

$$CDF(t) = 1 - e^{-\lambda t},\ CDF^{-1}(r) = \frac{-ln(r)}{\lambda} = i \tag{2}$$

A stationary arrival rate λ may be conceivable for some applications. However, in the case of translation between macro- and mesoscopic simulation a varying λ will be the rule. When λ changes over time, the

PAP is called non-homogeneous. Figure 8 shows an exemplary sampling of arrivals with varying arrival rate. Handling varying inputs from a macroscopic model, this has to be considered. We chose a communication mode, in which the macroscopic simulator only publishes an HLA interaction when a parameter of a border link has changed. A road's volume $L_I^V(r)$ translates directly to a corresponding λ. If the translator waited

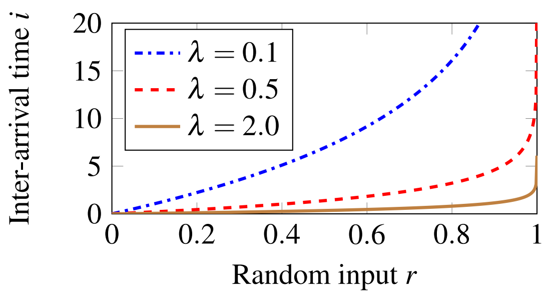

Figure 7: Inverse CDF.

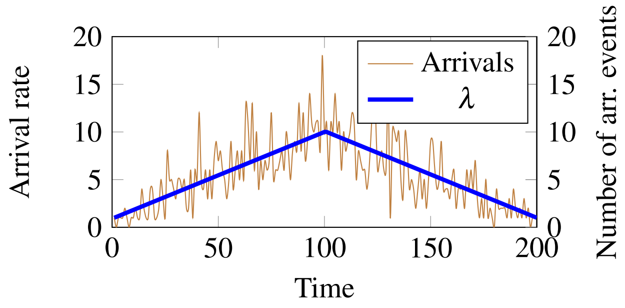

Figure 8: Sampled arrivals.

for the currently scheduled arrival event to happen before processing an updated arrival rate, this would lead to errors. Imagine that λ changes from 0.001 to 1 at time step $t = T_{NOW}$, while the first arrival was sampled with $\lambda = 0.001$ to step T_1. This would lead to $T_1 - T_{NOW}$ steps of waiting with a wrong arrival rate, in which on average $\lambda * (T_1 - T_{NOW})$ arrivals would have been sampled. Therefore, we record T_{SAMPLE}, the time when an arrival T_1 was sampled. On λ-change at T_{NOW}, we sample again and modify the newly sampled arrival time T_2 by the proportion of the already elapsed time $T_E = T_{NOW} - T_{SAMPLE}$ to the new inter-arrival time $T_D = T_2 - T_{NOW}$ to calculate the new arrival time $L_O^T(r) = T_1'$. This is shown in Equation 3 and depicted in Figure 9.

$$T_1' = T_{SAMPLE} + \frac{T_E}{T_D} * (T_1 - T_{SAMPLE}) + (1 - \frac{T_E}{T_D}) * (T_2 - T_{NOW}) \tag{3}$$

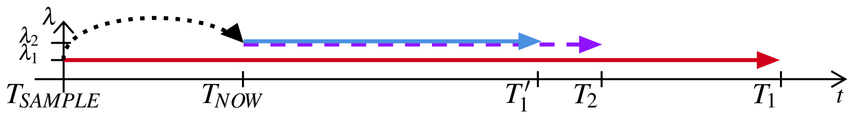

Figure 9: Sampling with a varying arrival rate.

The translator holds a list of roads and their current λ, next inter-arrival time, and time when the arrival was scheduled. This list is sorted by arrival times. Algorithm 1 shows that macro.road and macro.volume is mapped to meso.road, meso.vehicle, meso.route. Therefore, the requirement for a surjective mapping is accomplished. The algorithm is clarifying another point: While the mathematical formulation of our translation process (Equation 2) is clear and indisputable, the question of how to realize the logic in code is not. When implementing the formula, ordering and timing of commands do make a difference.

4.2.2 TU 1 B (Aggregator): From Meso to Macro

There are many possible ways of aggregating data from meso to macro. Again, the mesoscopic simulator will only communicate when an event happens. In the meso case, this means a vehicle enters a border link. The translator B also only communicates when inferred parameters differ from previously forwarded ones. A moving average is used to calculate a smoothed volume $L_O^V(r,t) = \bar{V}(r,t)$ for a road r from input events at fixed time points t (Equation 4). With a weight factor w, the moving average of the volume is calculated, where $C_{t_n}^r$ is the subset of cars in the set of input tuples L_I that were on road r since the last sample step. Considering that, a count of recent inputs per road needs to be saved on translator side. After

$\bar{V}(r,t)$ is calculated, the input count for r is set to zero. Surjective mapping is again given: meso.road and meso.vehicle transform to macro.road and macro.volume (Algorithm 2).

$$\bar{V}(r,t_n) = (w-1) * \bar{V}(r,t_n - 1) + w * |\frac{C^r_{t_n}}{t_n - t_{n-1}}|, \;\; C^r_{t_n} = \{c \in L_I | L^R_I(c,t_x) = r, \;\; t_{n-1} < t_x \leq t_n\} \tag{4}$$

```
Function loop():
    rd := roads.getFirstArrival();
    meso := {rd.roadId, rndName(), ringRoute};
    while simTime < rd.intArrTime+rd.sampleTime do
        if receivedInteraction then
            return;
    sendInteraction(meso);
    rd.intArrTime := arrival(rd.lambda);
    rd.sampleTime := simTime;
    roads.replaceAndSort(rd);
Function arrival(lambda):
    return-ln(rnd())/lambda;
Function varArrival(rd, lambda):
    intArrTime := arrival(lambda); $T_S$ := rd.sampleTime;
    $R$ := (simTime - $T_S$) / (simTime - $T_S$ + intArrTime);
    return $T_S$ + $R$*rd.intArrTime + (1-$R$)*intArrTime;
Function receive(macro):
    lambda := macro.volume/stepLength;
    rd := roads.get(macro.roadId); sampleTime = simTime;
    intArrTime := varArrival(rd,lambda) - sampleTime;
    rd := {rd.roadId, intArrTime, sampleTime, lambda};
    roads.replaceAndSort(rd);
Function send(meso):
    hla.sendInteraction(meso);
```

Algorithm 1: Sample arrivals from macro.

```
Function loop():
    foreach rd ∈ roads do
        vol := rd.vehicleCount/windowDuration;
        rd.oldVol := rd.vol; rd.vehicleCount=0;
        rd.vol := ($w-1$)*rd.vol + $w$*vol;
        if rd.vol != rd.oldVol then
            send({rd.roadId, rd.vol});
Function receive(meso):
    roads.get(meso.roadId).vehicleCount++;
Function send(macro):
    hla.sendInteraction(macro);
```

Algorithm 2: Average arrivals from meso.

4.3 TU 2: Mesoscopic and Microscopic

The transition between meso and micro does not face the problems of connecting discrete and continuous time simulation. Regardless, one may have to deal with different step lengths. Both models are seen from a inner view of the system, contrary to the macroscopic view. Therefore, to realize the transition, the mesoscopic parameter set for a vehicle can be extended with additional information and the microscopic parameter set will be cropped. As a consequence, the previously used `loop()` function (Algorithm 3, 4) is not needed. We can directly react when information is received. Such a reaction could also just consist of modifying internal counters, e.g. if translation happens between two different scaling levels.

4.3.1 TU 2 C (Disaggregator): From Meso to Micro

The event of an arrival occurs and does not need to be sampled like in the previous case. We assume a mesoscopic model that uses a queue-based approach to move a car c between roads. Additional information that has to be inferred is chosen lane l and velocity v. To estimate a velocity $L^V_O(c) = v$, some kind of normal speed v_n and standard deviation σ need to be predefined. Given that, we generate a random velocity from a Gaussian distribution G with a mean at $\mu = v_n$. Therefore, two numbers r_1, r_2 are drawn from a uniform distribution between -1 and 1 and the Marsaglia polar method (Marsaglia and Bray 1964) is used to sample a normally distributed velocity v (Equation 5). To estimate the lane id $L^L_O(c) = l = r_3 * NL * \frac{v}{v_n}$, we assume that faster drivers tend to choose a more centered lane and multiply the number of lanes NL with the proportion of v_n and v and a uniform random number r_3 between 0 and 1. To simplify, we ignore turning lanes. Road R and route RT can be taken directly from the input tuple: $L_O^R(c) = L_I^R(c), L_O^{RT}(c) = L_I^{RT}(c)$.

$$v = \sigma * r_1 * p + v_n, \quad p = \sqrt{\frac{-2 * ln(r_1^2 + r_2^2)}{r_1^2 + r_2^2}}, \quad 0 < r_1^2 + r_2^2 < 1 \tag{5}$$

4.3.2 TU 2 D (Aggregator): From Micro to Meso

The aggregation from micro to meso is the simplest of all four translations. When a micro tuple is received, a meso tuple (road id, vehicle id, route) is directly cropped out of the micro tuple and sent back to the federation bus: $L_O^R(c) = L_I^R(c), L_O^{RT}(c) = L_I^{RT}(c)$. There is no additional processing needed (Algorithm 4).

```
Function receive(meso):
    vel,lane := sampleVL();
    micro := {meso.roadId, meso.veh,
      meso.route, lane, vel};
    send(micro);
Function send(micro):
    hla.sendInteraction(micro);

Function sampleVelLane():
    while 0<q<1 do
        $r_1$ := rnd(); $r_2$ := rnd();
        q := $r_1$ * $r_1$ + $r_2$ * $r_2$ ;
    p := sqrt(-2*ln(q)/q);
    vel = $\sigma$ * $r_1$ * p + $v_n$;
    lane = rnd() * $NL$ * (v/$v_n$);
    return vel, lane;
```

Algorithm 3: Sample speed and lane from meso.

```
Function receive(micro):
    send({micro.roadId,
      micro.veh, micro.route});

Function send(meso):
    hla.sendInteraction(meso);
```

Algorithm 4: Crop micro to meso.

5 OVERHEAD ESTIMATION

Naturally, using reusable translation federates entails a timing overhead. As more federates will participate, the message volume will also increase. In this section, the overhead of detaching the translation units to separate federates in an HLA federated simulation is estimated. It is shown how much additional effort is spent in order to decide if the advantages of the proposed approach prevail. CERTI was used as open source HLA implementation (Noulard et al. 2009). The simulation scenario and the models themselves are of secondary importance here, since the focus is on the information transfer itself. To achieve an enduring information exchange between the different models, we randomly created a 966 m long circular road. 835 cars are spawned in the center of the micro simulator's area and drive one turn. Combined with the ring shape, this means that every traffic information needs to traverse all different translations and modeling paradigms. The scenario lasts for one hour with increasing traffic load and is also published (Gütlein 2019). The resulting federation is shown in Figure 10, while responsibilities are depicted in Figure 11.

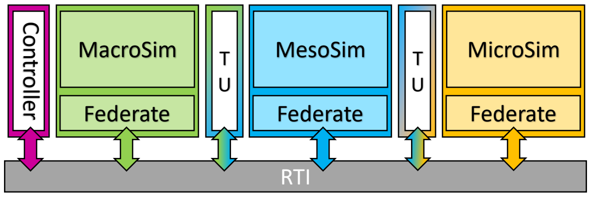

Figure 10: Federation with TUs.

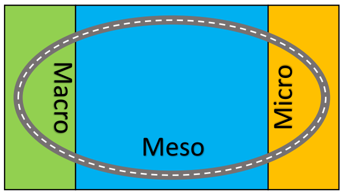

Figure 11: Topology.

We measured the needed computational time for every ten-second duration of simulation time. The resulting average effort per second of simulation time is plotted in Figure 12 for both approaches. Accordingly, we had four federates in the first case and six federates in the second case. We used MATSim as a mesoscopic simulator and SUMO as a microscopic simulator. The macroscopic simulator is self-written and based on the assumption that a link's incoming and outgoing volume are equal (w.r.t. the traversal time). MATSim and SUMO were run with a step length of 1 s. The timing overhead in the detached case is clearly visible. In total, it took 1.34 times the runtime of the classical approach. The volume of sent data over the federation bus is also higher. The total volume of sent messages is with ∼4.3 MB in the detached case 3.26 times higher than in the classic case (∼1.3 MB). As the ratio of simulation effort and data transfer covers a worst case scenario (due to the shape and the short length), the usual overhead can be expected to be even lower. Thus, the approach may still be applicable if multi-level simulation is used to gain performance.

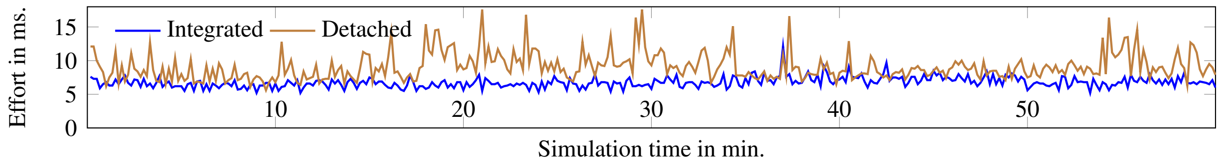

Figure 12: Time effort per second of simulation time.

6 CONCLUSION AND OUTLOOK

The paper presented an approach that empowers reusability in coupled simulation models, which is relevant for combinations of different (data) models. Apart from the conceptual attempt to use well-defined and detached translation units, and the methodical implementation by communicating via HLA, a concrete example on how to realize a multi-level traffic simulation was given. The concept offers a more transparent data flow between the submodels and therefore enhances reproducibility. Due to the modularization aspect, isolated testing of translation components, and comparison of translation algorithms is enabled. Through encapsulation into separate HLA federates, platform independence, programming language independence, and independence from a specific simulation tool's API is given for translation units. This improves the possibilities of reusing and exchanging single simulation components of a certain data layer.

The timing overhead was estimated with a traffic simulation. Thereby, the problem of modeling route choices out of aggregated traffic data was left out and the vehicles simply had to follow the road. Since this work is about the detachment concept itself, this issue is of no consequence. Besides, the timing and data transmission overhead can be seen as a drawback, especially if simulating as fast as possible is required or data transmission is expensive. In other cases, the advantages such as reusability (subsection 3.5) prevail.

Regarding future steps, many ideas are conceivable. The modular concept allows for extending the TU's interface. Extensions can involve logging, feedback loops for calibrating translation algorithms, concatenation of multiple TUs, or recording and replaying of information flow. Records enable validation at a later point and provide replay capabilities. This can be used to feed isolated parts of the whole model with information and re-simulate these parts with different simulation parameters. Analogously, real or simulated data from a different layer that comes from external sources (e.g. traffic counting stations) may be fed into a simulation using a TU. By concatenating multiple TUs, the amount of possible connections increases. Related to our traffic simulation example, this would allow the coupling of a macroscopic and a microscopic simulator without any additional logic, simply by connecting the macro-meso TU_1 and the meso-micro TU_2 to one macro-micro TU_3 that translates between the macro and the micro simulation.

REFERENCES

Balci, O., G. L. Ball, K. L. Morse, E. Page, M. D. Petty, A. Tolk, and S. N. Veautour. 2017. "Model Reuse, Composition, and Adaptation". In *Research Challenges in Modeling and Simulation for Engineering Complex Systems*, edited by R. Fujimoto, C. Bock, W. Chen, E. Page, and J. H. Panchal, 87–115. Cham: Springer International Publishing.

Burghout, W., H. Koutsopoulos, and I. Andreasson. 2005. "Hybrid Mesoscopic-Microscopic Traffic Simulation". *Transportation Research Record: Journal of the Transportation Research Board* (1934):218–255.

Cox, K. 1998. "A Framework-based Approach to HLA Federate Development". In *Proc. of the 1998 Fall Sim. Interop. Workshop*. September 14th-18th, Orlando, FL.

Dahmann, J., R. M. Fujimoto, and R. M. Weatherly. 1998. "The DoD High Level Architecture: An Update". In *Proc. of the 1998 Winter Simulation Conference*, edited by D. Medeiros et al., 797–804. Piscataway, New Jersey: IEEE.

Dalle, O. 2012. "On Reproducibility and Traceability of Simulations". In *Proc. of the 2012 Winter Simulation Conference*, edited by C. Laroque et al., 1–12. Piscataway, New Jersey: IEEE.

Eldabi, T., S. Brailsford, A. Djanatliev, M. Kunc, N. Mustafee, and A. F. Osorio. 2018. "Hybrid Simulation Challenges and Opportunities: A Life-Cycle Approach". In *Proc. of the 2018 Winter Simulation Conference*, edited by M. Rabe et al., 1500–1514. Piscataway, New Jersey: IEEE.

Gütlein, M., R. German, and A. Djanatliev. 2018. “Towards a Hybrid Co-Simulation Framework: HLA-Based Coupling of MATSim and SUMO”. In *Proc. of the 2018 IEEE/ACM 22nd International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, edited by E. Besada-Portas et al., 234–242. Piscataway, New Jersey: IEEE.

Gütlein, M. 2019. “detachedTranslation”. https://github.com/cs7org/detachedTranslation, accessed 01.04.2019.

Ibrahim, R., N. Regnard, P. L’Ecuyer, and H. Shen. 2012. “On the Modeling and Forecasting of Call Center Arrivals”. In *Proc. of the 2012 Winter Simulation Conference*, edited by C. Laroque et al., 1–12. Piscataway, New Jersey: IEEE.

Kang, H., Y. Sung, H. Kwon, S. Yoon, and S. Choi. 2018. “An Open Architecture Framework for Electronic Warfare Based Approach to HLA Federate Development”. *Security and Communication Networks* 2018:Article ID 5150730.

Klein, U., T. Schulze, and S. Straßburger. 1998. “Traffic simulation based on the high level architecture”. In *Proc. of the 1998 Winter Simulation Conference*, edited by D. Medeiros et al., Volume 2, 1095–1103. Piscataway, New Jersey: IEEE.

Marsaglia, G., and T. A. Bray. 1964. “A Convenient Method for Generating Normal Variables”. *SIAM review* 6(3):260–264.

Martinez-Moyano, I. J., and C. M. Macal. 2016. “A Primer for Hybrid Modeling and Simulation”. In *Proc. of the 2016 Winter Simulation Conference*, edited by T. Roeder et al., 133–147. Piscataway, New Jersey: IEEE.

Meyer, B. 1987. “Reusability: The Case for Object-Oriented Design”. *IEEE Software* (2):50–64.

Noulard, E., J.-Y. Rousselot, and P. Siron. 2009. “CERTI, an Open Source RTI, why and how”. In *Spring Simulation Interoperability Workshop*. March 23th-27th, San Diego, CA.

Oses, N., M. Pidd, and R. J. Brooks. 2004. “Critical issues in the development of component-based discrete simulation”. *Simulation Modelling Practice and Theory* 12(7-8):495–514.

Ozaki, A., M. Furuichi, N. Nishi, and E. Kuroda. 2000. “The Use of High Level Architecture in Car Traffic Simulations”. *IEICE Transactions on Information and Systems* 83(10):1851–1859.

Pawlikowski, K., H.-D. Jeong, and J.-S. Lee. 2002. “On Credibility of Simulation Studies of Telecommunication Networks”. *IEEE Communications magazine* 40(1):132–139.

Pos, A., P. Borst, J. Top, and H. Akkermans. 1996. “Reusability of Simulation Models”. *Knowledge-based systems* 9(2):119–125.

Radeski, A., S. Parr, and R. Keith-magee. 2002. “Component-Based Development Extensions to HLA”. In *Proceedings of the Spring Simulation Interoperability Workshop (SISO)*. March 10th-15th, Orlando, FL, Paper ID: 02S-SIW-046.

Ruscheinski, A., and A. Uhrmacher. 2017. “Provenance in Modeling and Simulation Studies - Bridging gaps”. In *Proc. of the Winter Simulation Conference*, edited by W. Chan et al., 872–883. Piscataway, New Jersey: IEEE.

Saulnier, E. T., and B. J. Bortscheller. 1994. “Simulation Model Reusability”. *IEEE Communications Magazine* 32(3):64–69.

Sewall, J., D. Wilkie, and M. C. Lin. 2011. “Interactive Hybrid Simulation of Large-scale Traffic”. *ACM Transactions on Graphics* 30(6):135:1–135:12.

Sung, C., and T. G. Kim. 2011. “Framework for Simulation of Hybrid Systems: Interoperation of Discrete Event and Continuous Simulators Using HLA/RTI”. In *Proceedings of the 2011 IEEE Workshop on Principles of Advanced and Distributed Simulation*. June 14th-17th, Nice, France, 54-61. Washington, DC: IEEE.

Taylor, S. J., T. Eldabi, T. Monks, M. Rabe, and A. M. Uhrmacher. 2018. “Crisis, What Crisis - Does Reproducibility in Modeling & Simulation Really Matter?”. In *Proc. of the 2018 Winter Simulation Conference*, edited by M. Rabe et al., 749–762. Piscataway, New Jersey: IEEE.

Tolujew, J., and F. Alcalá. 2004. “A Mesoscopic Approach to Modeling and Simulation of Pedestrian Traffic Flows”. In *Proc. of the 18th European Simulation Multi-Conference*. June 13th-16th, Magdeburg, Germany, 13-16.

Uhrmacher, A. M., S. Brailsford, J. Liu, M. Rabe, and A. Tolk. 2016. “Panel—Reproducible Research in Discrete Event Simulation—A Must or Rather a Maybe?”. In *Proc. of the 2016 Winter Simulation Conference*, edited by T. Roeder et al., 1301–1315. Piscataway, New Jersey: IEEE.

Yilmaz, L., S. J. Taylor, R. Fujimoto, and F. Darema. 2014. “Panel: The Future of Research in Modeling & Simulation”. In *Proc. of the 2014 Winter Simulation Conference*, edited by A. Tolk et al., 2797–2811. Piscataway, New Jersey: IEEE.

Zhu, F., Y. Yao, H. Chen, and F. Yao. 2014. “Reusable Component Model Development Approach for Parallel and Distributed Simulation”. *The Scientific World Journal* 2014(March):Article ID 696904.

Zhu, F., Y. Yao, J. Li, and W. Tang. 2019. “Reusability and Composability Analysis for an Agent-based Hierarchical Modelling and Simulation Framework”. *Simulation Modelling Practice and Theory* 90(2019, January):81–97.

AUTHOR BIOGRAPHIES

MORITZ GÜTLEIN received a diploma in computer science in 2017. He is a Research Assistant and PhD student at the Department of Computer Science 7, Friedrich-Alexander University Erlangen-Nürnberg (FAU). Besides vehicular communication and mobility patterns, his research interests include multi-level simulation and simulation performance. moritz.guetlein@fau.de.

ANATOLI DJANATLIEV is an Assistant Professor (AkadR) at University of Erlangen-Nürnberg. His research interests include various topics on simulation and modeling using SD, DES, ABS, hybrid simulation, and methods for healthcare decision-support. Most recently, he focuses on topics from the area of connected mobility and autonomous driving. anatoli.djanatliev@fau.de.