Gentle Coupling of Pedestrian Behavior Model Implementations: a Pedestrian Simulator Interoperability Protocol

Peter M. Kielar[1], Daniel H. Biedermann[2] und Felix Dietrich[3]

[1]Chair of Computational Modeling and Simulation · Technische Universität München · Arcisstraße 21 · 80333 Munich · peter.kielar@tum.de
[2]Chair of Computational Modeling and Simulation · Technische Universität München · Arcisstraße 21 · 80333 Munich · daniel.biedermann@tum.de
[3]Chair of Scientific Computing in Computer and Science · Technische Universität München · Boltzmannstraße 3 · 85748 Garching · and · Department of Computer Science and Mathematics · Hochschule München · Lothstrasse 64 · 80335 Munich · felix.dietrich@tum.de

Zusammenfassung

Die Simulation von Fußgängerströmen ist eine moderne Methode um Fußgängerverhalten computergestützt vorherzusagen. Die zeitgenössische Fußgängerforschung befasst sich fortlaufend mit der Entwicklung von neuen und ausgeklügelten hybriden Fußgängerverhaltensmodellen. Solche Modelle verbinden bestehende Fußgängerverhaltensmodelle in einem einzelnen Konzept. Leider hat sich gezeigt, dass beim Koppeln dieser Modelle nicht nur die Softwarearchitektur eines Personenstromsimulators zugeschnitten werden muss, was ein legitimier Ansatz ist, sondern auch eine vermeidbare Veränderung der Implementierung der Fußgängerverhaltensmodelle durchgeführt werden muss. Hier zeigen wir, dass unser entwickeltes Interoperabilitätsprotokoll es Forschern ermöglicht hybride Modelle in einem Fußgängersimulator zu integrieren ohne die originalen Implementierungen der Fußgängerverhaltensmodelle anzupassen.

Abstract

Pedestrian flow simulations are a modern method for computationally predicting pedestrian behavior. In contemporary research, the development of new and sophisticated hybrid pedestrian behavior models is ongoing. These models couple other pedestrian behavior models into a single concept. However, it was shown that a coupling of different models not only leads to alterations of the simulator software architecture, which is a legit approach, but also enforces an avoidable change of the pedestrian behavior model implementations. Here we show that our developed interoperability protocol enables researchers to integrate hybrid models in a pedestrian simulator without changing original behavior model implementations.

1 Introduction

Pedestrian simulations are an approach to forecast pedestrian behavior. The results of simulations help to assess threads to pedestrians beforehand in different environments. Pedestrian simulations implement pedestrian models in a computer program. Therefore, a computer sci-

ence background is mandatory to create sophisticated pedestrian simulation systems. In addition to pedestrian models, such systems consist of different software modules which are essential for running a pedestrian simulation, e.g., pedestrian handling modules, result writers, analysis algorithms, time step handler, and geometry modules.

We identified an additional component of pedestrian simulation systems that was no focused in research yet. This is an interoperability module that enables multiple pedestrian behavioral models to be smoothly coupled without changing the underlying model implementations. Pedestrian model coupling is an emerging research topic in pedestrian dynamics that inherits opportunities for improving pedestrian behavior forecasts by creating hybrid models. Despite the work done regarding the theoretical aspects of model coupling, the computational interoperability perspective is undeveloped. This results in inflexible pedestrian simulation architectures and large software implementation expenses. Here we provide a new middleware concept in form of an interoperability protocol for pedestrian model coupling that enables researcher to couple pedestrian models with low effort. The main advantage is that the original implementations of the coupled models stay unchanged. Therefore, the developed interoperability protocol saves time and improves flexibility in hybrid model development. The protocol enables to exchange pedestrian data on an arbitrary data channel. In general, the exchange of data and commands message passing is outsourced to a protocol handler module. The concept works for simulator networks as well as for models within a single simulator.

2 Related work and theoretical background

Pedestrian research regarding behavioral models received increasing attention in recent years. Walking models predict small scale walking behavior of pedestrians. Typical examples of such models are the social force model (HELBING ET AL. 2000), the gradient navigation model (DIETRICH & KÖSTER 2014), and cellular automata concepts (BLUE & ADLER 2001). Navigation models describe pedestrian routing behavior, thus a walking path adjoining a route network is forecast (GERAERTS & OVERMAS 2007; KNEIDL 2013). Strategical models predict the location the tactical model should find a route to (KIELAR ET AL. 2014). In contemporary research, new hybrid approaches are developed. It was shown that the coupling of existing models can improve the overall flexibility and feasibility of pedestrian simulations. The hybrid concept of KNEIDL ET AL. (2013), CHOORAMUN ET AL. (2013), and BIEDERMANN ET AL. (2014) are representative examples for hybrid models.

The rise of hybrid models creates new challenges for pedestrian dynamics researcher. The coupling of models does not only demands to define a conceptual coupling, but also creates computational interoperability requirements. Surprisingly, the software architecture related research of hybrid models is not approached in pedestrian dynamics. Architectural pedestrian simulator concepts exist (CURTIS ET AL. 2014; TOLL ET AL. 2015), but interoperability is not in focus. Hence, available pedestrian simulation architectures do not cover computational interoperability between arbitrary models in detail. We found modern approaches for simulation and model interoperability in work published regarding general simulation software architectures (DAHMANN ET AL. 2014) and simulation middleware concepts (AL-ZOUBI ET AL. 2011). Hence, we use state of the art computer science background as guideline for creating the interoperability protocol for pedestrian simulators. Additionally, modern approaches for coupling software modules by using process and protocol based methods (HOLZMANN 1991; SHETH 1999) are appropriately taken into consideration. For modelling

the protocol, we utilized well-established software modelling methods (HAREL 1987; RUMBAUGH 2004) and crafted an interoperability protocol for pedestrian models and simulators based on our fundamental knowledge regarding pedestrian model implementations.

3 Pedestrian interoperability protocol modelling

The interoperability protocol is similar to other packages and modules of a pedestrian simulator. Therefore, we describe how the protocol handler module fits to generic pedestrian simulator architectures and provides details of the protocol processes and data exchange concepts.

3.1 Pedestrian simulator integration

The protocol is a module implementable to pedestrian multi-simulator environments as well as inner pedestrian simulator environments. Therefore, an instance of the protocol handler can be either paired with a model in a simulator or a simulator in a simulator network. For both cases, an instance of the protocol handler in the system is defined as controller, and the other instances are labeled as clients. The user defines the components by configuration. In a system implementing the protocol, the clients exclusively exchange commands and data via the data channel; thus as long as the channel can carry the data transport burden, an arbitrary number of clients can be integrated. The system comprises configurations, locks, shared data, and simulation outputs. The locks are used for synchronizing and scheduling the clients' execution processes and the configurations provide initial information. The shared data container stores results created by the models temporarily. The simulation results contain the simulation output data of a model. We provide the top-down view on the protocol client and host structure in Figure 1 for one host and two clients. For clarification, the dashed arcs define the command flow between the components; the solid arcs model the simulation data flow.

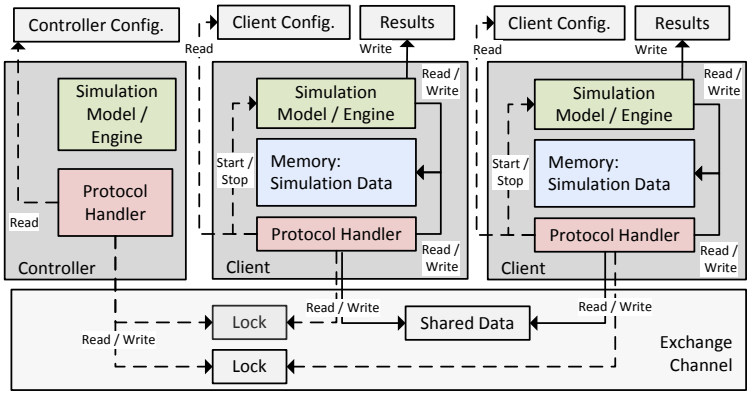

Fig. 1: The top-down model description of a generic simulation system implementing the interoperability protocol. The system may be in a single simulator or a network system; thus the Model / Engine labels may swap.

3.2 Interoperability protocol

The steps each protocol handler has to execute during start-up, execution, and shut-down process are defined in this Section. At the beginning of the start-up process, the controller reads the configuration and gets access to the client control-schedule. The schedule is defined as given in in Table 1. The clients' protocol handler should be initialized with a configuration consisting only of the entity block, which describes the client's own properties. It is important

to note that clients in the same parallel section do not have to have the same cycle runtime. In general, the multiplicity of client calls is only dependent on the configuration. Thus complex runtime systems can be crafted using this definition.

Table 1: Building blocks of the client control-schedule stored in the controller configuration in the extended Backus-Naur Form. Client configuration contains an entity element indication the clients own configuration only.

Schedule	::=	**Structure**
entity	::=	“(” “shared : id” “_” “lock : id” “_” “cycles : integer” “_” “end : integer “)” ;
parallel	::=	[parallel] | entity ;
sequence	::=	“{” [sequence] | parallel “}” ;

Figure 2 provides a state-chart diagram of the start-up and shut-down process of the protocol. Each protocol handler executes the process right at start-up. The execution procedure for the controller and the clients are presented as state-charts in Figure 3 and Figure 4. The controller process is in charge of handling the schedule; thus the execution of models by sequencing client processes. Both components check if the simulation has ended by tracking the simulation steps internally. Clients are in charge of handling the shared data and running a simulation cycle each time the controller allows execution, which is indicated by an existing lock.

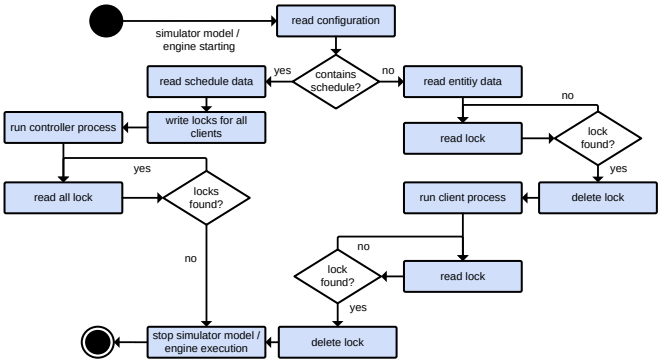

Fig. 2: The start-up and shut-down process for a protocol handler. The client and controller main processes are modelled as black boxes here.

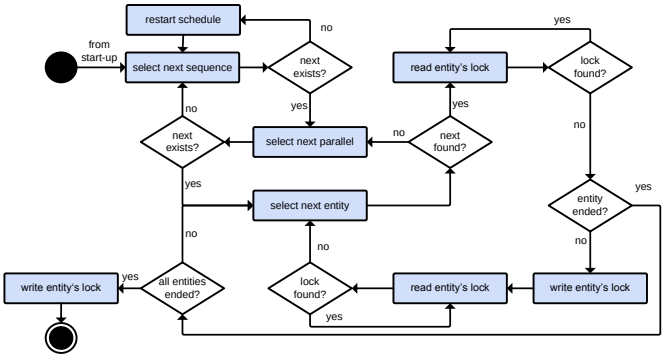

Fig. 3: The controller’s main execution process. This process is embedded in the “run controller process” of the start-up and shut-down state chart

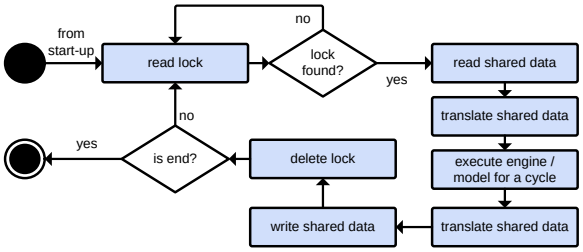

Fig. 4:
The client's main execution process. This process is embedded in the "run client process" of the start-up and shut-down state chart.

3.3 Exchanged pedestrian data

The data exchanged between the pedestrian simulation models or engines depends on the application. Thus, the hybrid model researchers have to define the exchangeable pedestrian related data. Hence, a basic requirement for utilizing the interoperability protocol is that the type definition can be translated to different simulation and model environments. A type table is a static lookup table which is given as additional configuration to the protocol handlers. The method for definition of a type table is given in Table 2. The lookup table enables the handler to translate data stored in the shared data container to the local data environment of the client. Zero entries indicate that a client cannot process the type and corresponding data. Thus, multiple clients can interact with partial knowledge on the same shared data. Still, non-obvious dependencies exist. Models or simulators running in parallel are executed in a sequence by the controller and are not allowed to change the same data sets entry. This knowledge is hidden within the clients and the validity of the data exchange has to be ensured within the hybrid pedestrian model concept.

Table 2: Building blocks of the pedestrian data lookup table stored in the protocol handler configuration. Shared data receivers are identified by the lock ids.

Types		**lock : id 1**	**lock : id …**	**lock : id n**
"1 : type"	≔	"0" | "1"	"0" | "1"	"0" | "1"
"… : type"	≔	"0" | "1"	"0" | "1"	"0" | "1"
"m : type"	≔	"0" | "1"	"0" | "1"	"0" | "1"

3.4 Proof of concept

As proof of concept, we utilized the protocol on the coupling of the simulators **MomenTum**, **VADERE**, and **TransiTUM**. **TransiTUM** (latin, "crossing") is a software tool that was developed at the Technische Unveristät München. It is a generic framework, which enables the multi-scale coupling of arbitrary pedestrian dynamics models (BIEDERMANN ET AL. 2014). **VADERE** (latin, "to go") is a Java based simulation software developed at the Munich University of Applied Sciences. It is capable of specification, simulation, 2D- and 3D-visualization, and analysis of microscopic models in pedestrian (e.g. DIETRICH & KÖSTER 2014) and car dynamics. Operation is possible through a graphical user interface or a console. **MomenTum** (latin, "movement force") is a cellular automata based microscopic pedestrian simulator integrating new concepts of pedestrian routing models developed at the Technische Unveristät München (KNEIDL 2013). The **MomenTum** software includes the 2D visualization and analysis tool SiNewVis for assessing and evaluation of pedestrian movement simulation.

Each simulator is running in a single executable; thus a simulator network is given. We added an additional component for taking the role of the controller. The component is referred to as **Demonstrator**. We implemented our interoperability protocol using the file system as data exchange channel for each component. Figure 5 provides the results of the successfully coupling of **MomenTum** and **VADERE** via **TransiTUM** utilizing the interoperability protocol. The configuration of the system is presented in Table 3 and the data exchange configuration is shown in Table 4.

Table 3: The configuration of the four components.

Component		**Configuration**
Demonstrator	:=	{(.\share.csv_.\t.lock_1_200)} {(.\share.csv_.\m.lock_1_100)}{(.\share.csv_.\t.lock_1_200)} {(.\share.csv_.\v.lock_1_100)}
TransiTUM	:=	(.\share.csv_.\t.lock_1 _200)
MomenTum	:=	(.\share.csv_.\m.lock_1_100)
VADERE	:=	(.\share.csv_.\v.lock_1_100)

Table 4: The data exchange table for all components.

Types	**“id : unique”**	**“x : double”**	**“y : double”**	**“v_x : double”**	**“v_y : double”**	**“v_d : double”**	**“target : unique”**	**“origin : unique”**
m.lock	“1”	“1”	“1”	“1”	“1”	“1”	“1”	“1”
t.lock	“1”	“1”	“1”	“1”	“1”	“1”	“1”	“1”
v.lock	“1”	“1”	“1”	“1”	“1”	“1”	“1”	“0”

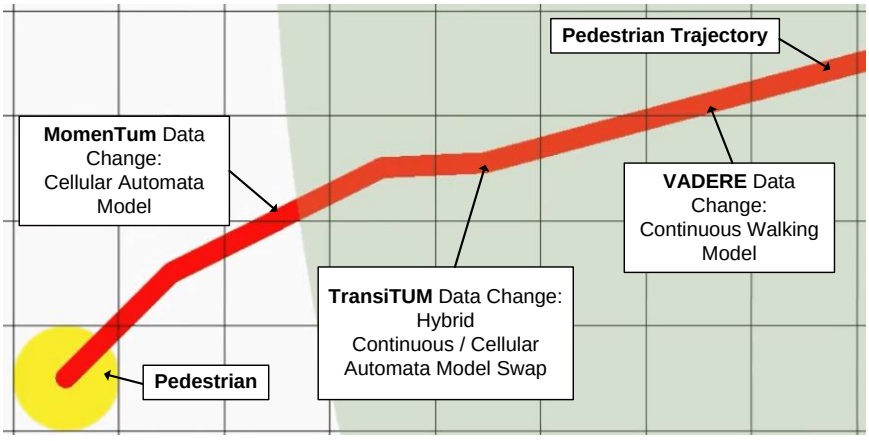

Fig. 5: A 2D visualization of the simulation results of **MomenTum**, **VADERE**, and **TransiTUM** running in a coupled environment using the interoperability protocol. The pedestrian data is exchanged between and changed by the models.

3.5 Feasibility analysis

We modelled the lock system part of the start-up, processing, and shut-down process of a client and a controller using a Petri-Net. Figure 6 presents the Petri-Net crafted with PIPE, a free Petri-Net modelling and analysis tool (DINGLE ET AL. 2011). We used PIPE to analyze the net and found that our mutual exclusive execution of the system works without deadlocks, is invariant, and finite. The number of sequences and the number of cycles are calculated based on the configurations by number of simulation blocks "end : integer" divided by the length of a simulation block "cycle : integer" without reminder.

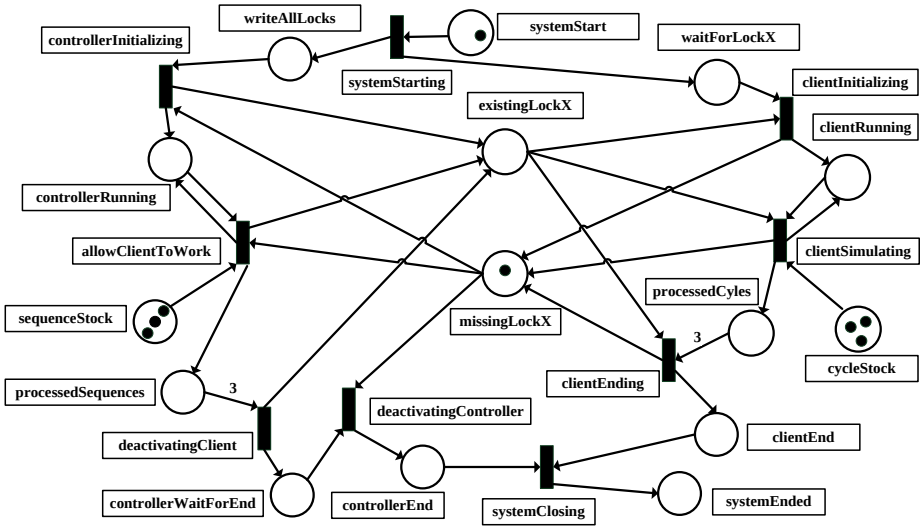

Fig. 6: The Petri-Net we created for a client and a controller, which can be extended to an arbitrary number of clients. The net proves that the system works as described in Section 3.2 regarding the crucial lock read and write operations.

4 Conclusion

Hybrid pedestrian models describe a merge of existing pedestrian models to create more sophisticated approaches. The model coupling demands a smooth integrating of the concepts in a pedestrian simulator without changing the underlying model implementations.

We created a new interoperability protocol that is utilized as middleware in a pedestrian simulator or a pedestrian simulator network. The protocol handles data exchange and the mutual exclusion execution of any number of clients. We proofed by a Petri-Net that our model is finite and dead-lock free. Additionally, we implemented the protocol successfully in a hybrid model environment; thus provide evidence that the protocol is applicable.

We target to improve the concept in future research by a creating a real parallel execution of the models or engines that is not based solely on sequencing. In general, the research on modern pedestrian simulator software architectures is promising since pedestrian dynamics is in need of efficient simulators that carry the burden to run complex models with high computational efficiently and little development effort.

Literature

AL-ZOUBI, K., & WAINER, G. (2011). Distributed simulation using restful interoperability simulation environment (rise) middleware. In *Intelligence-Based Systems Engineering* 129-157. Springer Berlin Heidelberg.

BIEDERMANN, D. H., KIELAR, P. M., HANDEL, O., & BORRMANN, A. (2014). Towards TransiTUM: A generic framework for multiscale coupling of pedestrian simulation models based on transition zones. *Transportation Research Procedia*, 2, 495-500

BLUE, V. J., & ADLER, J. L. (2001). Cellular automata microsimulation for modeling bi-directional pedestrian walkways. *Transportation Research Part B: Methodological*, *35*(3), 293-312.

CHOORAMUN, N., LAWRENCE, P. J., & GALEA, E. R. (2011). Implementing a hybrid space discretisation within an agent based evacuation model. In *Pedestrian and Evacuation Dynamics* (pp. 449-458). Springer US.

CURTIS, S., BEST, A., & MANOCHA, D. (2014). Menge: A modular framework for simulating crowd movement. *University of North Carolina at Chapel Hill, Tech. Rep*.

DAHMANN, J. S., & MORSE, K. L. (1998, JULY). High level architecture for simulation: An update. Distributed Interactive Simulation and Real-Time Applications. In *Proceedings 2nd International Workshop on*, 19(20), 32-40.

DIETRICH, F., & KÖSTER, G. (2014). Gradient navigation model for pedestrian dynamics. *Physical Review E*, *89*(6), 062801

DINGLE, N. J., KNOTTENBELT, W. J., & SUTO, T. (2009). PIPE2: a tool for the performance evaluation of generalised stochastic Petri Nets. *ACM SIGMETRICS Performance Evaluation Review*, *36*(4), 34-39.

GERAERTS, B., & OVERMARS, M. (2007). The corridor map method: a general framework for real-time high-quality path planning. *Computer Graphics and Image Processing*, 18(2), 107. doi:10.1002/cav.166.

HAREL, D. (1987). Statecharts: A visual formalism for complex systems. *Science of computer programming*, *8*(3), 231-274.

HELBING, D., FARKAS, I., & VICSEK, T. (2000). Simulating dynamical features of escape panic. *Nature*, *407*(6803), 487-490.

HOLZMANN, G. J. (2007). Design and Validation of Computer Protocols.

KIELAR, P. M., HANDEL, O., BIEDERMANN, D. H., & BORRMANN, A. (2014). Concurrent Hierarchical Finite State Machines for Modeling Pedestrian Behavioral Tendencies. *Transportation Research Procedia*, *2*, 576-584.

KNEIDL, A. (2013). Methoden zur Abbildung menschlichen Navigationsverhaltens bei der Modellierung von Fußgängerströmen, Technische Universität München.

KNEIDL, A., HARTMANN, D., & BORRMANN, A. (2013). A hybrid multi-scale approach for simulation of pedestrian dynamics. *Transportation research part C: emerging technologies*, *37*, 223-237.

RUMBAUGH, J., JACOBSON, I., & BOOCH, G. (2004). *Unified Modeling Language Reference Manual, The*. Pearson Higher Education.

SHETH, A. P. (1999). Changing focus on interoperability in information systems: from system, syntax, structure to semantics. In *Interoperating geographic information systems*, 5-29. Springer US.

VAN TOLL, W., JAKLIN, N., & GERAERTS, R. (2015). Towards Believable Crowds: A Generic Multi-Level Framework for Agent Navigation