Towards Semantic Composition of Event-Based Simulation

Sandro Koch
sandro.koch@kit.edu
Karlsruhe Institute of Technology

Abstract

Extensible and maintainable designed software architecture allows creating sustainable and robust software systems. Instead of programming each variant of an architecture to analyze different quality aspects simulation is used. Simulation allows coping with many different architectures without losing time and money in actually implementing each individually. But simulations usually are not developed to be extensible or maintainable. In this paper, we describe problems that arise when simulations are developed in a modular way and propose an approach how these problems can be addressed.

1 Introduction

In the field of event-based simulation development, monolithic programming is widely used. Although event-based simulations are decoupled regarding time, space and synchronization [3] adding new functionality to an existing simulation can create unnecessary effort in terms of maintainability, extensibility, and reusability. This effort is the result of dependencies between the interacting elements of a simulation. Each element which needs to transport information must structure the message as the receiver expects it to be encoded. Therefore, the sender needs specific knowledge about each receiver to whom information is sent. This results in a high coupling between the elements that have to exchange information, making maintenance and reuse difficult. However, splitting a simulation up into smaller sub-simulations could improve the maintainability regarding a more manageable project size. By creating a simulation from dedicated modules, each module can be reused in a different simulation. Problems that occur when modularizing a simulation are shown in [9] which corresponds to Fig. 1. We will be concentrating on the coupling of simulation modules and the preservation of behaviour as mentioned in [9]. Combining modules of different simulations or even different domains require that modules can be combined and produce expected results.

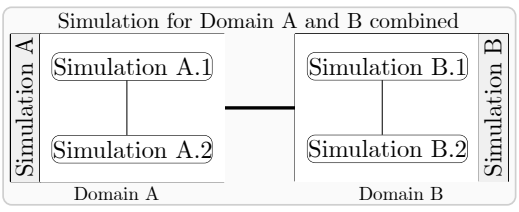

Figure 1: Inter- and Intra-Simulation Coupling

As shown in Fig. 1 we define a modularized simulation as a combination of simulation modules. It can contain multiple modules, and each module can include sub-modules. Coupling takes place across the simulation of different functionalities (i.e., A and B) or within one functionality (i.e., A.1 and A.2 or B.1 and B.2). The communication between modules is realized with events and it is represented as relation between the simulation modules (i.e., between A and B, A.1 and A.2, B.1 and B.2).

2 Semantic Composition

At the semantic level, the type of a simulation module and the type of an event must be distinguishable to determine whether and how these modules can be coupled. In order to be able to couple simulation modules as required, we discuss the basic but necessary requirements for the structure of the simulation modules and the event structure.

2.1 Simulation Module Interfaces

In [8] different kinds of simulations are presented. We categorize simulation modules into consuming modules, providing modules and a combination of both. For instance, a consuming module would be a monitor which only receives data but does not participate in the simulation itself. A typical providing module would be a load driver that generates load that is not affected by the simulation results.

Thus, the coupling of a simulation module can be described by two interfaces, one to define the structure of the required information and one to define the structure of the information provided. These interfaces are defined by the types of events that can be sent or received by a simulation module. This makes it possible to couple simulation modules based on their interface specification without knowing the actual implementation. This means that individual modules can be replaced and, if necessary, better maintained.

2.2 Event Structure

An event must be distinguishable regarding time, source, destination and type. For a monolithic simulation, it is not relevant how strongly the coupling between the elements in a simulation is realized. But as soon as the simulation is divided into modules, not any module can be combined. Time is needed to determine when an event was generated and if necessary, in which incoming order events have to be processed. In a modular simulation, the sender (source) must be known, i.e., if an answer has to be sent. Also the

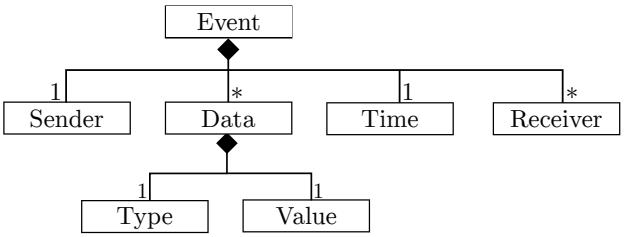

Figure 2: Model of an event

receiver must be specified so that the event triggers the right simulation module. If a module can receive different types of events, the type must be defined separately. Fig. 2 illustrates which information of an event is needed to enable coupling of individual simulation modules. An event contains one sender and one-time stamp. To reach multiple receivers, the event can include more than one receiver. Also, multiple data fields with different types are possible.

3 Limitations

Our approach in modularizing simulations by defining interfaces for different kinds of modules and specifying events have to be evaluated. Monolithic event-based simulations need to be modularized, coupled and then checked if the behaviour is equal to the monolithic simulation. Besides the pure functionality, the aspect of performance changes regarding the simulation execution has to be investigated too. Although a simulation usually runs at design time to allow profound decision making, the execution time can be relevant if the computation of one simulation needs much more time.

4 Related Work

Modularization is an established topic in the software engineering community. Using modules on an architectural level [7] or on code level as in [6] are some examples. A composability lexicon published by [4] clears the connotation of composability in the context of simulations. The Distributed Interactive Simulation (DIS) [1] standard realized the composability of simulations on the protocol level. Based on the DIS standard, the High-Level Architecture (HLA) [2] evolved. The CODES approach of [5] utilizes an ontology approach to model a discrete event simulation. These proposed approaches are very restrictive regarding the standards they use, and none have an approach for semantic coupling of simulation modules.

5 Conclusion and Future Work

In this paper, we presented our vision for developing simulation modules that can be coupled and used for creating modular simulations. Our discourse comprises the discussion of defining the interfaces of a module on a generic level. Furthermore, we define which elements are crucial to model events that suite the combinability of the described modules. The proposed structure is designed to make simulations reusable and better maintainable. We are currently working on a metamodel of modular simulations that will make it possible to describe and implement simulations. The metamodel is used to modularize existing event-based simulations and to make a statement about how modularization affects the performance of a simulation.

Acknowledgement

This work was supported by the Ministry of Science, Research and the Arts Baden-Württemberg in the funding line Research Seed Capital (RiSC).

References

[1] D. S. Committee et al. “The DIS vision: A map to the future of distributed simulation”. In: *Institute for Simulation and Training* (1994).

[2] J. Dahmann et al. “The department of defense high level architecture”. In: *Conference on Winter simulation*. IEEE, 1997, pp. 142–149.

[3] P. T. Eugster et al. “The Many Faces of Publish/Subscribe”. In: *ACM Computing Surveys* 35.2 (2003), pp. 114–131.

[4] M. D. Petty and E. W. Weisel. “A composability lexicon”. In: *Spring Simulation Interoperability Workshop*. Vol. 2003. 2003, pp. 181–187.

[5] Y. M. Teo and C. Szabo. “CoDES: An integrated approach to composable modeling and simulation”. In: *Simulation Symposium*. IEEE, 2008, pp. 103–110.

[6] R. Jung et al. “GECO: A Generator Composition Approach for Aspect-Oriented DSLs”. In: *ICMT*. Springer, 2016, pp. 141–156.

[7] R. H. Reussner et al. *Modeling and simulating software architectures: The Palladio approach*. MIT Press, 2016.

[8] R. Heinrich et al. “Integrating business process simulation and information system simulation for performance prediction”. In: *Software & Systems Modeling* 16.1 (2017), pp. 257–277.

[9] S. Koch. “Challenges in Modularization of Discrete Event Simulations”. In: *Collaborative Workshop on Evolution and Maintenance of Long-Living Software Systems*. CEUR-WS, 2018.