Multi-level Hardware Simulation in Palladio

Sebastian Weber
sebastian.weber@fzi.de
FZI Forschungszentrum Informatik

Bahareh Taghavi
bahareh.taghavi@kit.edu
Karlsruhe Institute of Technology

Abstract

Predicting the fulfillment of quality requirements, e.g., performance, is important during design and implementation of a software system to ensure its implementation can meet these requirements. Model-based analysis is a common approach to get such predictions. It usually requires a trade-off between the accuracy of the result and the execution time of the analysis, because resources for the execution of the analysis are limited, but some components might require a more accurate analysis. When looking at traditional model-based analyses, like the Palladio approach, the user can decide this trade-off and with it the level of granularity of the model and simulation only prior to, but not throughout, the simulation execution. In addition, the complexity and heterogeneity of systems is increasing which complicates modelling and analysis on a single level further. This paper presents a concept for the extension of the Palladio approach to support modelling and analysis of systems on multiple levels of abstraction to overcome these limitations.

1 Introduction

Software systems are growing in size and complexity. Broy et al. [2] mention an exponential increase and estimate that more than 80% of the innovations in the automotive domain come from computer systems. Besides software this includes advancements in specialized and general purpose hardware. Alongside the development of manually written code, the amount of functionality based on artificial intelligence, like emergency brake or lane keeping assistant and the hardware required to run it is rising, according to Bello et al. [6]. This leads to complex and heterogeneous systems that are difficult to analyse with traditional model-based analyses, because both software and hardware have different behavior and properties regarding the fulfillment of quality requirements. A single analysis capable of analysing such systems would have maintainability and extensibility issues due to the large design space it has to cover. Also, the execution time of an analysis on a low level of abstraction for the whole system would not be feasible.

To overcome the challenges emerging from this scenario, (1) a model-based analysis should support modelling and analysis on multiple levels of abstraction, and (2) the decision between execution time and accuracy of the analysis should be changeable throughout the analysis. Additionally, (3) it should be composed of independent existing modelling and analysis tools to improve maintainability and ensure a focus on extensibility. The first two challenges (1) and (2) can be addressed by multi-level simulation. This concept, according to Ghosh [1], enables simulation at more than one level of abstraction, e.g., the gate- and circuit-level in the embedded domain. Other more recent examples from other domains are the MUSA approach from Grass et al. [3] for high performance computing or by D'Angelo et al. [5] for the internet of things. Both approaches provide a multi-level simulation specific for their domain and composed as a single analysis with fixed levels of abstraction.

The drawbacks of these current approaches is their limited reusability and maintainability, because they are designed to solve a specific use case from a specific domain. To achieve the third goal (3), we aim at providing a framework to compose multiple hardware modelling and analysis tools. In this paper we describe a specific case for using such a framework, namely the coupling between the the software architecture simulator Palladio approach and a low level hardware simulator. While this achieves (3) it results in a more complex composition and it induces more overhead for the communication between the simulators then the examples for multi-level simulation mentioned above. The contribution of this paper is a concept for the composition of two independent hardware simulators to form a multi-level hardware simulator composed with Palladio. This includes the following subcontributions:

- In contrast to existing approaches, we discuss and apply required composition operators, because we compose independent simulators.
- We present a concept for switching between two independent simulators instead of a specialised algorithm integrated in a multi-level simulation.
- We give an overview on required information in the Palladio Component Model (PCM) to support the two levels of abstraction in this concept.

The following section 2 introduces the relevant parts of Palladio, section 3 presents the outlined contribution and the conclusion summarizes the paper in the last section 4.

2 Palladio

The Palladio approach [4] is used to predict the fulfillment of quality requirements, for example performance, of component-based software systems based on five different models. These models form a PCM together, which is used as input for the analysis of the system. The parts of the PCM relevant for this paper are resource demands and resource containers. Resource demands specify how much of a resource is necessary to execute a certain method and which resource is needed. They are specified on a high level of abstraction, e.g., as number of cycles for the resource Central Processing Unit (CPU) and file size for an Hard Disk Drive (HDD). It is important to note, that these specifications are only examples for the notion of resource demands. Resource containers model hardware components. They consist of one or multiple resources with a corresponding processing rate like processor frequency or read and write speed and can be wired together with other resource containers. Based on the PCM, the Palladio approach simulates the execution of software on hardware under a specified usage load. We replace the current hardware simulator part of Palladio with the multi-level hardware simulator. Therefore, usage and software simulation remains unchanged.

3 Multi-level Hardware Simulation

Composition operators Extending the Palladio approach with a multi-level hardware simulation requires two steps of analysis coupling. Heinrich et al. [7] present different analysis composition operators from which we use the two explained in the following. The first is composition by result exchange. It can be used when information exchange is required only between simulation executions and not throughout them. The results of one simulation can then be used as input for another one. When using composition by co-simulation simulators can communicate and exchange information in real-time throughout a simulation. The presence of a coordinator is required for synchronization and information exchange.

The first composition necessary is the composition of multiple hardware simulators by result exchange to form a multi-level hardware simulator. This multi-level hardware simulator takes a specific level of abstraction and a corresponding model as input and simulates the execution of the model with the simulator of the specified level. As Figure 1 shows, the concept described here supports two levels with Palladio's hardware simulation as the first one with a focus on execution time (*Palladio hardware simulation*) and a hardware simulator with a focus on accuracy (*Hardware simulation*). The composition by result exchange enables the subsequent execution of single simulators which ensures that the execution time of the multi-level simulation is as low as possible. Running multiple simulators concurrently would result in the execution time of the slowest simulator being the overall execution time of the multi-level simulator. The second analysis composition is between the multi-level hardware simulator and Palladio. It requires the concurrent execution and synchronization of both parts and uses composition by co-simulation.

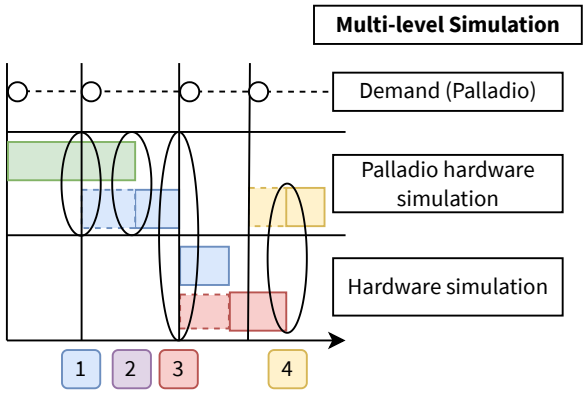

Figure 1: Events during the multi-level hardware simulation

Switch between simulators Figure 1 and the corresponding Table 1 show the four different event types that can occur during the multi-level hardware simulation described in this paper involving two concurrent demands. The x-axis of Figure 1 represents time, the y-axis the different analysis tools. The demands are shown as circles on the level of *Demand (Palladio)* with vertical lines. The colored rectangles in the levels of *Palladio hardware simulation* and *Hardware simulation* represent the simulation of the execution of those demands. The level at which each demand should be analysed is specified before the depicted simulation execution. The ellipses mark the four different event types.

These event types describe conditions during the execution of the multi-level simulation and how to react to them. An event is triggered either by a new demand or the completion of a demand. The following demand can either be on the same or a different level. Events involving only one demand are omitted for reasons of space. Depending on the current state of the multi-level hardware simulation, a different event type occurs. For simplicity, we made the two following assumptions about the system: (1) First Come, First Served Scheduling (2) Maximum of two concurrent demands. The second assumption (2) is justifiable, because increasing the number of concurrent demands does not lead to additional event types, due to the chosen scheduling strategy. Table 1 summarizes the types of events that can occur during the simulation.

The first two types of events (1, 2) deal with demands on the same level of abstraction. Hence, there is no switch between simulators. Due to the chosen

Event	Demand	Description
1	New Demand	Same level
2	Demand finished	Same level
3	New Demand	Low to high level → switch once new demand known
4	Demand finished	High to low level → switch not until demand finished

Table 1: Events, demand and description

scheduling strategy, the simulation of the execution of the second blue demand is postponed until the first demand is finished, which is marked by the dashed border. The third event type (3) is a new demand on the lower level of abstraction. As soon as the new red demand is scheduled, the level of abstraction is switched. This enables the lower level hardware simulator to compute a suitable hardware state by simulating parts of the blue demand. The last type (4) is the switch to a higher level simulation on demand simulation completion. Because the red demand should be simulated on the lower level, the simulator can not be switched until it has been completed.

Required information In addition to the simulators, the models have to be considered on both levels too. Parts of the PCM have to be modelled differently for lower and higher level hardware simulators. Table 2 shows some simplified model elements related to the CPU of a single resource container that are required by the concept explained above. If the choice which level to use for the simulation of a demand is static, this information is also required in the PCM.

Element	Palladio hardware simulator	Hardware simulator
Resource demand	No. of cycles	Executable code for the chosen system
Hardware description	No. of cores & their frequency	Hardware architecture of the processor, No. of cores & their frequency, size & associativity of L1 & L2 caches, size & structure of memory, etc.

Table 2: Model elements required on both levels

When switching between different hardware simulators, the state of the simulator switched in has to be approximated. Using an empty or default state is only viable if the simulator has enough time to compensate for this inaccuracy. When switching from a lower to a higher level, the state of the higher level simulator can be generated from the lower level simulator. Going the other way is in contrast not possible, because lower level information can only be estimated. Therefore additional tools are necessary, e.g., functional simulators or emulators, to gather the required information.

4 Conclusion

In this paper, we presented a concept for the usage of multi-level hardware simulation in the Palladio approach. The concept is motivated by challenges derived from software development in the automotive domain and distinguished from other works in the field of multi-level simulation. We explained the required composition operators and elaborated on the multi-level hardware simulator by showing when and how a switch between hardware simulators is done, which parameters are needed and how the state of simulators can be generated. We plan to implement the presented concept and derive insights for the planned domain-independent framework for multi-level simulation.

Acknowledgements

This work was funded by the DFG (German Research Foundation) – project number 499241390 (FeCoMASS) and "Kerninformatik am KIT (KiKIT)" funded by the Helmholtz Association (HGF).

References

[1] S. Ghosh. "On the concept of dynamic multi-level simulation". In: *Proceedings of the 19th annual symposium on Simulation*. 1986, pp. 201–205.

[2] M. Broy et al. "Engineering automotive software". In: *Proceedings of the IEEE* 95.2 (2007), pp. 356–373.

[3] T. Grass et al. "MUSA: a multi-level simulation approach for next-generation HPC machines". In: *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE. 2016, pp. 526–537.

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

[5] G. D'Angelo, S. Ferretti, and V. Ghini. "Multi-level simulation of internet of things on smart territories". In: *Simulation Modelling Practice and Theory* 73 (2017), pp. 3–21.

[6] L. L. Bello et al. "Recent Advances and Trends in On-Board Embedded and Networked Automotive Systems". In: *IEEE Transactions on Industrial Informatics* 15.2 (Feb. 2019), pp. 1038–1051.

[7] R. Heinrich et al. "Challenges in the Evolution of Palladio—Refactoring Design Smells in a Historically-Grown Approach to Software Architecture Analysis". In: *Composing Model-Based Analysis Tools*. Springer International Publishing, 2021, pp. 235–257.