Towards Multi-Level-Simulation in Dynamic Cloud Environments

Stefan H. A. Wittek[1], Michael Göttsche[2], Andreas Rausch[1] and Jens Grabowski[2]
[1] *Department of Informatics, Clausthal University of Technology, Clausthal-Zellerfeld, Germany*
[2] *Institute of Computer Science, Georg-August-University Göttingen, Germany*
switt@tu-clausthal.de, michael.goettsche@cs.uni-goettingen.de, grabowski@cs.uni-goettingen.de, arau@tu-clausthal.de

Keywords: Co-Simulation, Cloud Deployment, Multi-Level-Simulation

Abstract: The engineering of cyber physical systems requires holistic simulation perspectives. The need of sufficient detail with regards to modelling resolution and simulation semantics in this perspective leads to increasing computation times and hardware costs required for these simulations. We aim to address this issue by providing a more efficient solution on these two layers. The system is simulated efficiently with regards to the model complexity. A holistic perspective is reached on a rough level, which is co-simulated with multiple detailed models of only those parts of the system that are of particular interest to the investigated phenomena. The resulting Multi-Level-Simulation is deployed in a dynamic cloud environment to use the provided hardware resources in a cost-efficient manner.

1 MOTIVATION

Cyber physical systems (CPS) consist of numerous hardware and software components. Autonomous cars and automated production facilities are examples of such systems. The task of designing CPS is difficult and has high risks, because of the complex, possibly unknown interdependencies between these components. In engineering, simulation has become a core method to evaluate designs, reduce the costs and effort resulting in build prototypes. The application of simulation to components of CPSs to evaluate and improve these components according to some criteria beneficial for the whole system is a common solution for the design of the CPS. Such a local approach may not lead to ideal systems, because crucial parts of their complexity lays in the interactions between their components.

To overcome this, much work is put into approaches that provide one holistic simulation of all components of such systems. Some relevant questions require high detailed models using computationally complex simulation semantics to be answered. Examples for this are the stiffness or the thermal behaviour of key structure elements of a design. To reflect this in a holistic approach, the entire system must be simulated in this high detail. All components need to provide sufficient detail and must be simulated using the same computationally intensive semantic and tool. This is not efficient, since this detail is not required in all components to answer particular questions. It would be sufficient to use abstracted models and simpler simulation semantics for these less relevant components.

We propose a simulation methodology and framework that simulates the CPS on multiple levels of abstraction at the same time. On the rough level, the whole CPS is modelled using a fast semantic. To answer questions that require more detailed simulations, only relevant components of the system are chosen to be simulated on more detailed levels.

In a traditional computing infrastructure setting, the resources have to be designed for the worst case, i.e. to satisfy the requirements of the most resource-intensive possible simulation run in order to generate its results in an acceptable timeframe. This poses no problem for simulations with homogeneous requirements. However, for cases where the resource utilization is highly heterogeneous, as in the case of our simulation methodology, the computing infrastructure that accommodates the worst case is vastly oversized for the average simulation, resulting in a low overall utilization and thus cost inefficiency.

A better choice for the computing infrastructure of this use case is one that allows to reserve and

release resources on-demand so as to dynamically match the requirements of the simulation. The Cloud Computing paradigm that has emerged and matured in the last few years matches this need. Thus, we will propose a framework for deploying simulations on a Cloud platform in order to achieve a timely as well as cost-efficient solution.

2 RELATED RESEARCH

A variety of works focus on the co-simulation of different simulations tools. Examples of this are the High Level Architecture specification for simulation interoperability (Dahmann et al., 1997), the Functional Mockup Interface standard for model exchange and co-simulation (Blochwitz et al., 2012) and the Mosaik Simulation API (Schütte et al., 2011). Another approach is to integrate different simulation semantics into a single tool. The Ptolemy project is an example for this approach (Eker et al., 2003). All this works aim towards a holistic simulation of the system under investigation. The simulation of different abstraction levels is only addressed in terms of tool integration. The task to provide proper interfaces to connect simulation on different levels has to be done by the modeller.

Much effort is put into approaches that provides such concepts for specific domains such as material flows (Dangelmaier and Mueck, 2004; Huber and Dangelmaier, 2011), traffic (Claes and Holvoet, 2009) or agent based behavior simulation. They center on the dynamic switching of abstraction levels of model parts at runtime. To do so, explicit mappings between the states of different levels are provided. These mappings are tightly bound to the domain and the simulations they connect and are not designed to be generalizable.

Some research is conducted investigating more general concepts for the problem. The approach of Dynamic Component Substitution describes a co-simulation as a set of connected software components communication through given interfaces (Rao, 2003). Switching a part of the simulation to a more detailed version corresponds to substituting one such component with another. Both components are required to have the exactly same interfaces. This is a critical limitation. If the components are situated on different levels of abstraction, it is plausible to expect the same for their interfaces. Multi Resolution Entities (Reynolds,Jr. et al., 1997) define a mapping that is used to synchronize the simulation state on different levels. These mappings are defined as invertible to use them in both directions. This requirement is only meet, if no information is lost mapping a detailed state to a rough state, which does not apply in general, as we will describe in section 2. The concept of Multi Resolution Modelling Space introduces adapters between the interfaces and several mappings between the states of simulations on different levels (Hong and Kim, 2013). However the problem of information loss is not addressed in this approach.

Our approach of Multi-Level-Simulation is different to these approaches, because it does not force the engineer to tailor the rough level simulations into components connected by interfaces. We consider this approach as too inflexible. The rough level can be modeled with no dependency to the detailed level. In fact, even cutting arbitrary parts out of existing rough level simulations to be linked to a detailed level is possible. The detailed simulations are linked into a single simulation on the rough level using only a state synchronization mechanism. This mechanism also addresses the problem of information loss.

Our approach towards dynamic deployment of the simulation infrastructure is novel as it addresses a use case for a dynamic simulation infrastructure that is more concerned with starting and stopping compute resources on-demand depending for the launched and cancelled simulation processes than with scaling long-running processes.

The topic of scaling computing infrastructures in Cloud environments for elastic applications has received a lot of attention. A predestined use case is that of scalable web applications, but more recently the research has shifted to scientific applications. In a thorough review (Lorido-Botran et al., 2015), the authors give an overview of the various auto-scaling techniques that have been addressed so far. Our own previous work has dealt with the question of acquiring compute resources and automating simulation deployment and execution for a statically sized infrastructure (Göttsche et al., 2015).

Research has also been done in the field of provisioning infrastructure for traditional simulation workflows. One proposal describes a service-oriented binding strategy including a middleware architecture for deploying simulation components (Vukojevic-Haupt et al., 2013). Recently, the TOSCA modelling standard has received more attention as a possibility for automating simulation workflows in a Cloud environment in a model-based way (Qasha et al., 2015). This has also been employed in research that proposes a domain-model-based deployment and execution framework for scientific applications (Glaser, 2015).

2 MULTI-LEVEL-SIMULATION

To describe our approach of multi-level-simulation in more detail, we consider the toy-example of a lift. Figure 1 shows the structure of this example.

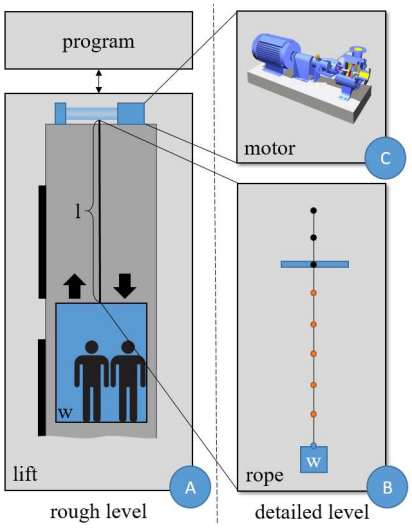

Figure 1: Structure of the elevator example.

(A) On the rough level it consists of a simulation modelling the structure of the lift and a lift program. The structure consists of a shaft in which a cabin can move. The cabin is rigged to a rope. The weight of the cabin (w) is altered when it stops at one of the exits. A motor manipulates the length of the rope (l). The program simulation is connected to the structure and handles the speed and direction of the motor. In this setup all parts of the structure are modelled as rigid bodies. The program has no sensor for l and positions the cabin only indirectly using the last position of the cabin and a timer. On this level, realistic scenarios of use are modelled . An example of this would be a whole day cycle of a office building. Most persons want to go up in the morning and down in the evening. The simulations on the level are fast.

(B) During the development of the lift and its program the engineers want to investigate, if the stretching of the rope caused by the weight of the cabin and the aging of the rope will lead to a wrong positioning of the cabin. To do this, a detailed but computationally intensive simulation of the rope is activated. This simulation is stateful to reflect the aging of the rope. Only parts of the rope that are stretched in a particular time step age. If the misplacement is a problem, the engineer has to implement a compensation of the phenomena in the program.

(C) After this, the dynamics of the cabin are investigated closely. A computationally intense simulation of the motor is activated. This simulation models the acceleration of the motor and allows to precisely determine the travel times of the lift. The simulation is stateful to model the heating of the motor which influences acceleration. Because the stretching of the rope is considered irrelevant for this question, the rope simulation is deactivated.

In both cases, parts of the lift are simulated on two levels at the same time. This leads to the challenge of maintaining the consistence between the states of both levels. If for example in (A) l is increased by 0.1m, all elements of the rope in (B) must be placed 0.1m lower. If in (B) the rope is stretched by 10%, displacing the lowest point from -3m to -3.3m, l must be set from 3m to 3.3m in (A).

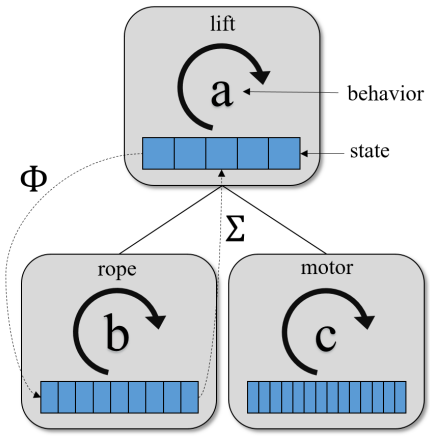

Figure 2: The problem of state synchronisation.

Figure 2 shows a schematic overview of the example. Each simulation consists of two parts. The state of a simulation is defined as a valuation of a fixed set of attributes. The behaviour of a simulation is defined as a mapping which has this state as input and output. Applying the behaviour to a state s_0 of a simulation leads to the state s_1. This corresponds to a step in the simulation. For all simulations the time Δt elapsing in one step is the same. The rough simulation of the lift is linked to a number of detailed simulation. Note that in the lift example only one of these simulations is connected in a particular simulation run.

Because the simulation model (i.e. the rope) are different, the attributes valuated in a state are different. The states need to be converted between the simulations. This is done using the state mappings Φ and Σ. At the current state of our work, these mappings are given by the modeller. Σ maps the detailed level state to the rough level state. It is typically not reversible, because information is lost. Referring to the lift example, there are a number of

different positions and age levels of the rope elements that map to the same *l*. Φ maps the rough level state to the detailed level. In the example, Φ restores the position of the rope elements using only *l*. To do so, Φ has to choose among a possible infinite set of states that are mapped to *l* by Σ. To account for this problem, we propose Φ as a mapping of the rough state and the last state of the detailed state.

Figure 3 gives an overview of the execution of the example. Note that in general changes on different levels accrue concurrently, regarding to simulation time.

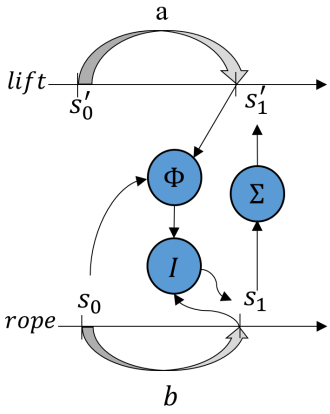

Figure 3: Execution of the lift example.

Let us consider the lift simulation starts with the initial state s'_0 and the rope simulation with the initial state s_0. The states are chosen so that $\Sigma(s_0) = s'_0$. This can be seen as that s_0 and s'_0 represent 'the same' on both levels. Now both simulations step using the behaviour function *a* and *b*. The rope simulation ages a number of rope elements, stretching the rope by 0.1m leading to the state s_1. In the same time step, the lift simulation unwinds the rope by 0.2m according to the initial speed of the motor, leading to the state s'_1. Converting s'_1 to a state of the rope simulation using Φ results in an intermediate state $\widehat{s_1}$. This state is in conflict to s_1 which was calculated using the behaviour *b* of the rope simulation. Simply overwriting s_1 using $\widehat{s_1}$ would annihilate the unwinding of the rope. To avoid this, an integrator function *I* must be employed to merge the two states. The resulting state contains both changes. Using Σ on this state leads to an integrated state of the lift simulation that contains again both changes. This state finally overwrites the state of the lift.

3 CLOUD-DEPLOYMENT

In this section, we describe aspects of deploying Multi-Level-Simulations in Cloud environments. Section 3.1 introduces elasticity aspects of Multi-Level-Simulations. In Section 3.2 we present an initial deployment of our prototype application. Finally, in Section 3.3, we outline our plan for a dynamic deployment strategy.

3.1 Elasticity in Multi-Level-Simulations

Multi-Level-Simulations are characterized by their variable resource requirements depending on the simulation question. The fluctuations result from the dynamic nature of MLS on two different layers:

- System level: A Multi-Level-Simulation consists of multiple components of which not all are operating at the same time. The required components vary (a) between different simulation runs depending on the simulation objective (b) within the same simulation run when components have finished their simulation.
- The component level complexity of a simulation is dependent on the simulation parameters. While a particular component's computational requirements may be low for one run, it can be higher for another.

In such cases, Cloud Computing can help in establishing a dynamic infrastructure to scale the resources in accordance with the simulations' demand. Ideally, at any point only the required computing resources will be allocated. Too few allocated resources ("underprovisioning") will lead either to longer runtimes or even abortion of the simulation. Too many resources ("overprovisioning"), on the other hand, allow for a timely execution of the simulation, but at the cost of dissipation.

3.2 Initial Deployment

As an initial approach for a Cloud deployment ~~for~~ of our prototype application we chose a static setup as depicted in Figure 3.

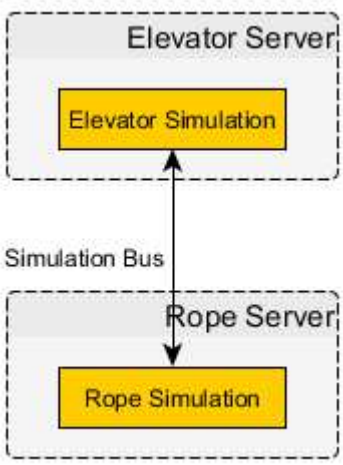

Figure 3: Initial Deployment

In this setup, each of the two simulation components is served by its own virtual machine in the Cloud. The components exchange status via RMI and therefore only require a shared network for communication.
On a technical level, our static deployment is model-based and agnostic to a particular Cloud platform. Concretely, we use the TOSCA-based Cloud orchestration platform Cloudify[1] as well as the Software Configuration Management tool Ansible[2]. This allows for a deployment of the simulation on the variety of Cloud platforms supported by Cloudify as well as on different operating systems as supported by Ansible. More importantly, it will serve as foundation for a dynamic deployment approach as outlined in the next section.

3.3 Dynamic Deployment

Our initial, static deployment lays part of the foundation for the intended dynamic deployment. However, the latter is more complex as its task, contrarily to the static case, is not finished once the required resources have been allocated and the components have been installed and launched on it. Instead, a framework that fulfils the three following tasks needs to be put in place:

- Monitoring: In order to perform runtime adaptations, the framework needs to collect information about the simulation resources and components. Specifically, the utilization of the resources is important to support a judgement.
- Reasoning: Using rules and the data collected by the monitor, the framework has to perform reasoning about infrastructure adaptation.
- Infrastructure Adaptation: The framework needs to adapt the infrastructure to the simulation requirements in both directions, i.e. by reducing or increasing its size. Also, it needs to adapt the deployment accordingly.

Specifically, for the reasoning task we will employ machine learning techniques that constantly refine the infrastructure adaptation. The reasoning pipeline is depicted in Figure 4. The computational complexity of a simulation depends on its model as well as its execution parameters. By combining this with information about the resource usage it is possible to build an execution history that serves as input for the reasoning engine for predicting a suitable deployment for future simulation runs.

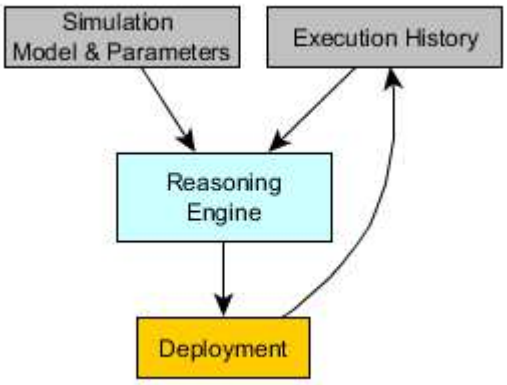

Figure 4: Reasoning Pipeline

For the monitoring and infrastructure adaptation tasks we intend to implement an integrated solution following a models@runtime (Aßmann et al., 2015) approach and the tools we employed for the initial deployment. This approach will allow a strong

[1] Online: http://getcloudify.org/
[2] Online: https://www.ansible.com/

decoupling of the adaptation logic from the technical steps necessary for enacting deployment changes.

4 STATUS AND FUTURE WORK

To get first insights on our concept of Multi-Level-Simulation and the corresponding cloud deployment mechanism, we build a prototype of the described lift example. The implementation is not connected to simulation tools but is based on simplified simulation stubs programmed in Java. The prototype consists of the lift and the program on the rough level and the rope on the detailed level. In the current state of the prototype, the mappings Φ, Σ and I are hand coded for the example. First results of this Multi-Level-Simulation are promising. The simulations stay in synchronisation and the results of the simulation meet our expectations.

The deployment of our prototype is distributed, but currently static. Concretely, the lift and the rope component are each deployed on their own virtual machine and the communication between the components is handled by our prototypical simulation bus which is based on Java's RMI. While still in an early stage, this bus will in the future be capable of handling a dynamically deployed simulation.

The provided lift example is useful to get first insights on the correctness of our method, but will be replaced by a real world example in order to provide validated results. As a next step, we will connect proper simulation tools to our prototype and implement a realistic example. Using hand coded mapping functions is not ideal in this case. We aim for a solution that is at least partially automated. To achieve this, a generic integrator function I, which is suitable for a variety of Multi-Level-Simulations, is researched. Employing machine learning algorithms to generate Φ, Σ seams promising. An input for such an approach could be a set of scenarios in which the same happens on both levels.

Another further direction will be the dynamic activation of detailed simulations at runtime. The rough level could be executed on its own, until an interesting state is reached. The detailed simulation is connected and is active only as long as needed.

Our next steps with regard to the dynamic Cloud deployment will include the evaluation of suitable strategies for the reasoning pipeline. Concretely, we will evaluate more realistic applications built with simulations tools to extract parameters that are informative for determining a simulations' resource requirements. Then we will assess their accuracy and build an integrated framework for dynamic deployment.

ACKNOWLEDGEMENTS

We thank the Simulationswissenschaftliches Zentrum Clausthal-Göttingen (SWZ) for financial support.

REFERENCES

Aßmann, U., Bencomo, N., Cheng, B. H., France, R. B., 2015. Models@ run. time (dagstuhl seminar 11481). *Dagstuhl Reports*, *1*(11).

Blochwitz, T., Otter, M., Akesson, J., Arnold, M., Clauss, C., Elmqvist, H., Friedrich, M., Junghanns, A., Mauss, J., Neumerkel, D., 2012. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models.

Claes, R., Holvoet, T., 2009. Multi-model Traffic Microsimulations, in: Winter Simulation Conference, WSC '09. Winter Simulation Conference, Austin, Texas, pp. 1113–1123.

Dahmann, J.S., Fujimoto, R.M., Weatherly, R.M., 1997. The Department Of Defense High Level Architecture.

Dangelmaier, W., Mueck, B., 2004. Using Dynamic Multiresolution Modelling to Analyze Large Material Flow Systems, in: Proceedings of the 36th Conference on Winter Simulation, WSC '04. Winter Simulation Conference, Washington, D.C., pp. 1720–1727.

Eker, J., Janneck, J.W., Lee, E., Liu, J., Liu, X., Ludvig, J., Neuendorffer, S., Sachs, S., Xiong, Y., others, 2003. Taming heterogeneity-the Ptolemy approach. Proc. IEEE 91, 127–144.

Glaser, F., 2015. Towards Domain-Model Optimized Deployment and Execution of Scientific Applications in Cloud Environments. Proceedings of the Doctoral Symposium at the 5th Conference on Cloud Computing and Services Sciences (DCCLOSER 2015), Lisbon, Portugal.

Göttsche, M., 2015. The 8th IEEE International Conference on Service Oriented Computing & Applications, Rome, Italy.

Hong, S.-Y., Kim, T.G., 2013. Specification of multi-resolution modeling space for multi-resolution system. SIMULATION 89, 28–40. doi:10.1177/0037549712450361

Huber, D., Dangelmaier, W., 2011. A Method for Simulation State Mapping Between Discrete

Event Material Flow Models of Different Level of Detail, in: Proceedings of the Winter Simulation Conference, WSC '11. Winter Simulation Conference, Phoenix, Arizona, pp. 2877–2886.

Lorido-Botrán, T., Miguel-Alonso, J., Lozano, J. A., 2012. Auto-scaling Techniques for Elastic Applications in Cloud Environments. Technical Report: University of the Basque Country, 11 – 14. doi:10.1145/2611286.2611314

Qasha, R., Cała, J., Watson, P., 2015. Towards Automated Workflow Deployment in the Cloud using TOSCA Towards Automated Workflow Deployment in the Cloud using TOSCA.

Rao, D.M., 2003. Study of Dynamic Component Substitutions (Dissertation). University of Cincinnati.

Reynolds,Jr., P.F., Natrajan, A., Srinivasan, S., 1997. Consistency Maintenance in Multiresolution Simulation. ACM Trans Model Comput Simul 7, 368–392. doi:10.1145/259207.259235

Schütte, S., Scherfke, S., Tröschel, M., 2011. Mosaik: A framework for modular simulation of active components in Smart Grids, in: Smart Grid Modeling and Simulation (SGMS), 2011 IEEE First International Workshop on. IEEE, pp. 55–60.

Vukojevic-Haupt, K., Karastoyanova, D. Leymann, F.: On-demand Provisioning of Infrastructure, Middleware and Services for Simulation Workflows. In: Proceedings of SOCA 2013