Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

SIMPLIFIED SIMULATION INTEROPERABILITY USING THE COCOBASIM APPROACH

Jörg Henss
Karlsruhe Institute of Technology
Karlsruhe, Germany
henss@kit.edu

ABSTRACT

When composing simulations from multiple domains, developers can choose from a long list of possible solutions. However, creating a functional and valid composition from existing simulation building blocks can be cumbersome. Existing solutions are often limited to specific platforms or require extensive and complex implementations. The *Coupled and Component-based Simulation* (CoCobaSim) approach aims at simplifying the development of an interoperable simulation composition using state-of-the-art model-driven techniques. It uses a component-based approach and employs interaction contracts to define simulation interactions. Based on the modeled information and a chosen simulation platform, developers can choose from several patterns and tactics to generate platform specific interoperability adapters and a suitable execution workflow.

1 INTRODUCTION

The theoretical problems of model composition and simulation interoperability have been tackled for years and many solutions have been proposed (Tucker and Gross 2013, Petty et al. 2014). Still, simulation developers tend to develop interoperable simulations over and over again. One reason is that no universally applicable solutions have been developed so far. Moreover, many proposed solutions are restricted to specific modeling formalisms and specific platforms or require a lot of additional implementation effort.

Interoperable simulations are commonly composed using the concept of simulation building blocks that encapsulate behavioral knowledge that shall be reused. This knowledge is often persisted in (semi-formal) models and executable code. When assembling a simulation from existing building blocks, problems are often caused by a lack of information on used concepts, parameter units and expected interaction sequences of these simulation building blocks. Furthermore, no information on the validity and pristine objectives of executable models is usually found in the definition of interfaces. This makes it difficult for developers to define a valid, credible and interoperable simulation composition (Law 2009).

Our approach is primarily aimed at supporting developers in the definition of interoperable simulation executions. This is especially difficult for semi-formal and purely code-based simulation models. The approach therefore supports additional patterns to enable interoperable simulations. These patterns are adjoined by interoperability tactics that can be chosen on component instance level.

2 APPROACH

The CoCobaSim approach combines a platform independent simulation component model with sophisticated model-driven techniques to simplify the development of interoperable simulations. A conceptual level simulation component model is used to define domain entities and *Simulation Component Types* reflecting a selection these entities. Furthermore, developer can use a textual DSL to define *Interaction Contracts* that specify interaction points of component types (Helm et al. 1990). These artifacts are domain specific and can be reused for other simulations in the same domain.

Developers assign component types and interactions to existing and newly developed simulation building blocks, thus creating *Simulation Component Instances*. While modeled component types are

platform independent, these component instances are platform dependent. There can be multiple component instances resembling the same component type, e.g., for representing different levels of granularity or different platforms. Component instances can refine the interaction contracts and define constraints on the exchanged data. This mechanism is also used to define the parameter ranges that have already been validated. Component instances are assembled to a system composition of coupled components. During this mapping step, syntactic and semantic gaps have to be identified and can be later on resolved.

Interoperability Patterns are the key concept of CoCobaSim that define how components and contracts are mapped to a specific execution on a chosen simulation platform. Examples for patterns are federations, iterative exchange and formalism-transformations. Furthermore, each pattern is accompanied with multiple *Adaptation Tactics*. Developers commonly choose and instantiate tactics on component level. Examples for tactics are the usage of gateway and proxy objects, model-transformations, and statistical-sampling methods.

Based on the chosen tactics and the previously defined component assembly, *Component Adapters* are generated using model-driven techniques. These adapters are used to bridge technological and semantic gaps, e.g. by adding missing parameters, creating additional events or doing unit conversion. In some cases only stubs can be generated that developers have to implement manually. When using statistical sampling tactics, these adapters encapsulate the statistical models.

Furthermore *Contract Guards* for checking the interaction contracts can be generated. These guards check for valid property ranges of exchanged entities and can implement final state machines for detecting invalid interaction sequences. If contract violations are detected, a warning is issued that a possible interoperability problem was detected. As some of the generated artifacts require manual addition of implementation details, a component repository stores these artifacts. Based on stored informations on the specific domain and platform, components can be reused in further compositions.

An *Interoperability Workflow Model* that allows defining interoperable simulation workflows complements the approach. The workflow model incorporates the actions derived from the chosen patterns and tactics and is generated semi-automatically. Moreover, it can include manual input, repeated and conditional actions, e.g. for deciding when an iterative fixed-point simulation has reached a stable state. Simulation results are persisted in a result repository alongside the used setup for enhancing reproducibility and credibility of results.

3 CONCLUSION

We presented the CoCobaSim approach for simplifying the development of interoperable simulations. It uses model-driven techniques to generate artifacts that ease the definition of valid compositions from existing simulation components.

In the future we aim at extending the supported set of platforms and add further interoperability patterns and tactics. On top of that, reverse-engineering approaches can be used to derive entity and contract information from existing code-based models. Finally, we are currently doing some studies on the applicability of the approach in different settings.

REFERENCES

Helm, R., I. M. Holland, and D. Gangopadhyay. 1990. *Contracts: specifying behavioral compositions in object-oriented systems*, Volume 25. ACM.

Law, A. M. 2009. "How to build valid and credible simulation models". In *Simulation Conference (WSC), Proceedings of the 2009 Winter*, 24–33. IEEE.

Petty, M. D., J. Kim, S. E. Barbosa, and J.-J. Pyun. 2014. "Software Frameworks for Model Composition". *Modelling and Simulation in Engineering* 2014.

Tucker, W. V., and D. C. Gross. 2013. "What More Do We Want in Modeling and Simulation Interoperability and Reuse?". GCMS '13, 27:1–27:8. Vista, CA.