Tutorial: Variantenmodellierung und automatisierte Simulationsexperimente

Thorsten Pawletta*, Hendrik Folkerts, Christina Deatcu

Forschungsgruppe Computational Eng. & Automation, Hochschule Wismar, Philipp-Müller-Str. 14, D-23966 Wismar, Germany; *thorsten.pawletta@hs-wismar.de

Abstract. Das Tutorial führt anhand einer Fallstudie in die Variantenmodellierung und Automation von Simulationsexperimenten auf Basis eines erweiterten *System Entity Structure/Model Base* (SES/MB) Ansatzes ein. Neben der Theorie und der Softwarearchitektur werden freie SES/MB-basierte Softwarewerkzeuge und deren Integration mit OpenModelica und MATLAB/Simulink vorgestellt.

Einleitung

Technische Systeme, die heute unter Schlagworten wie Internet der Dinge, Cyber-Physical-Systems oder Industrie 4.0 zusammengefasst werden, sind durch eine hohe Komplexität und Variabilität gekennzeichnet. Im Software-Engineering (SE) definieren Capilla et al. [1] Variabilität als die Fähigkeit, ein System je nach Einsatzzweck und Zielstellung zu konfigurieren, zu erweitern oder anzupassen. Demgemäß ist mit Variabilität zumeist eine hohe Variantenvielfalt verbundenen. Zur Beherrschung vielfältiger Softwarevarianten wurde im SE, in Analogie zu Produktlinien in der Fertigung, das Konzept der *Software Product Lines* (SPLs) entwickelt. Das SPL-Engineering umfasst Methoden und Werkzeuge zur Erstellung unterschiedlicher Softwarevarianten auf Basis einer gemeinsamen Plattform.

Wie im SE stellt das Management von Variabilität und Variantenvielfalt eine Herausforderung im Bereich der Modellbildung und Simulation (M&S) dar. Das Pendant zu SPLs im SE sind Modellfamilien in der M&S. Nach Zeigler [2] ist eine Modellfamilie, die Menge aller Modelle der Entwurfsalternativen, die sich unter Beachtung der gewählten Systemgrenzen und der verfolgten Entwurfsziele ergeben. Die konkrete Herausforderung besteht demgemäß: (i) in der Modellierung einer Modellfamilie, (ii) der Auswahl eines oder mehrerer Kandidatenmodelle, (iii) der Generierung eines oder mehrerer ausführbarer Simulationsmodelle und (iv) der Bewertung der Simulationsergebnisse. Das Tutorial zeigt wie, aufbauend auf dem von Zeigler [3] eingeführten und durch verschiedene Kollegen kontinuierlich weiterentwickelten Ansatz der *System Entity Structure / Model Base* (SES/MB) ([4], [5]), die gestellten Anforderungen gemeistert werden können. Darüber hinaus wird gezeigt, wie simulatorunabhängige MBs aufgebaut werden können und wie auf Basis einer erweiterten SES/MB-Architektur automatisierte Experimente mit Modellfamilien durchführbar sind.

Im Tutorial steht nicht die Vollständigkeit der einzelnen SES/MB-Methoden mit allen Erweiterungen im Vordergrund. Es geht primär um das grundlegende konzeptionelle Verständnis und die prinzipiellen Möglichkeiten zur Automation von Simulationsexperimenten mit Modellfamilien, einschließlich der Verwendung unterschiedlicher Simulatoren.

Aufbau des Tutorials

Das Tutorial führt schrittweise anhand eines Fallbeispiels in die Variantenmodellierung und Automation von Simulationsexperimenten auf Basis eines erweiterten SES/MB-Ansatzes ein ([6], [7]). Nebenläufig zur Theorie wird das Fallbeispiel schrittweise implementiert. Dabei wird auf freie SES/MB-basierte Softwarewerkzeuge eingegangen und deren Integration mit Simulationsumgebungen am Beispiel von OpenModelica und MATLAB/Simulink gezeigt. Die freien Werkzeuge stehen unter GitHub zur Verfügung ([8], [9]). Das Tutorial ist in sieben Schwerpunkte unterteilt und endet mit einem Fazit.

Fallbeispiel. Es werden zwei mögliche Strukturvarianten eines Regelsystems eingeführt. Es gilt, für unterschiedliche Regelziele und Nebenbedingungen die möglichst minimale Struktur mit dazugehöriger Parametrierung per Systemsimulation zu finden.

SES/MB-basierte Modellierung. Es werden die grundlegenden Prinzipien der SES/MB-basierten

Modellierung von Modellfamilien eingeführt. Danach werden am Fallbeispiel die simulatorunabhängige Spezifikation von Modellvarianten, Modellstrukturen und Modellparametrierungen mit einer SES diskutiert. Anschließend wird für das Fallbeispiel eine simulatorabhängige MB unter Verwendung von dynamischen Modelica Basiskomponenten aufgebaut.

Implementierung einer SES. Ausgehend von der für das Fallbeispiel spezifizierten SES werden ein Python-basierter SES-Editor und dessen Integration mit einem separaten SES-Viewer gezeigt. Dabei werden fortgeschrittene SES-Methoden praktisch demonstriert, wie zum Beispiel das Mergen von SES-Modellen. Für den Austausch mit anderen Werkzeugen werden SES-Modelle im JSON- oder XML-Dateiformat gespeichert.

Modellauswahl und -generierung. Es wird die Ableitung von konkreten Modellkonfigurationen aus einer SES und die Generierung von simulatorspezifischen Modellen gezeigt. Es folgt eine softwaretechnische Demonstration für das Fallbeispiel, wobei ausführbare Modelle für OpenModelica generiert werden.

Die freie MATLAB SES Toolbox. Aufgrund der weiten Verbreitung von MATLAB/Simulink wurden spezielle SES/MB-basierte Werkzeuge entwickelt, welche als APP in das System MATLAB/Simulink integrierbar sind. Am Fallbeispiel werden unter Verwendung der speziellen Tools alle Schritte von der Modellierung der simulatorunabhängigen SES sowie dem Aufbau einer MB mit Simulink bis zur Generierung und Ausführung von Simulink-Modellen aufgezeigt.

Organisation simulatorunabhängiger MBs. Basierend auf dem Functional-Mock-up-Interface (FMI) wird ein Ansatz zum Aufbau simulatorunabhängiger MBs aufgezeigt und am Fallbeispiel softwaretechnisch demonstriert.

Automation von Simulationsexperimenten. Es wird eine erweiterte SES/MB-Architektur mit Komponenten zur vollständigen Automation von Simulationsexperimenten vorgestellt. Ausgehend von einer Experimentspezifikation mit den Entwurfszielen und geltenden Nebenbedingungen werden: (i) zulässige Modellkonfigurationen aus der SES abgeleitet, (ii) simulatorspezifische oder FMI-basierte Modelle generiert, (iii) mit einem Simulator ausgeführt und (iv) die Simulationsergebnisse analysiert. In Abhängigkeit der erzielten Ergebnisse werden weitere Modellkonfigurationen studiert oder das Experiment beendet. Anhand des Fallbeispiels wird die Automation softwaretechnisch demonstriert.

Fazit. Abschließend werden bestehende Defizite und mögliche Weiterentwicklungen des SES/MB-Ansatzes sowie der Software-Tools im Kontext der aktuellen Forschung diskutiert.

Danksagung

Wir danken Bernie Zeigler und Jerzy Rozenblit für die wertvollen Diskussionen und die Förderung unserer Forschung.

References

[1] Capilla R, Bosch J, Kang K-C, editors. *Systems and Software Variability Management: Concepts Tools and Experiences.* Heidelberg, New York et al.: Springer, 2013.

[2] Zeigler B.P, Praehofer H, Kim T.G. *Theory of Modeling and Simulation.* 2nd Ed. San Diego: Academic Press; 2000.

[3] Zeigler B.P. *Multifaceted Modeling and Discrete Event Simulation.* 1st Ed. London: Academic Press; 1984.

[4] Rozenblit J.W, Zeigler B.P. Representing and constructing system specifications using the system entity structure concepts. In *Proc. of Winter Simulation Conference;* 1993; Los Angeles, CA, USA, 604-611.

[5] Zeigler B.P., Sarjoughian H.S. Guide to Modeling and Simulation of Systems of Systems. London: Springer Pub.; 2013

[6] Pawletta T, Durak U, Schmidt A. Modeling and Simulation of Versatile Technical Systems Using an Extended System Entity Structure / Model Base Infrastructure. In: Zhang L, Zeigler B.P, Laili Y editors. *Model Engineering for Simulation*. 1st Ed. London: Academic Press / Elsevier; 2019. 393-418.

[7] Folkerts H, Pawletta T, Deatcu C. Model Generation for Multiple Simulators Using SES/MB and FMI. In *25th ASIM Symposium Simulationstechnik (SST);* 2020 Okt; virtuell. ARGESIM Report 59. 13-20, doi: 10.11128/arep.59.a59003

[8] Folkerts H. *Python-based SES/MB Toolset*. https://github.com/hendrikfolkerts

[9] CEA Wismar. *SES Toolbox for MATLAB*. https://github.com/cea-wismar/SES_Tbx_Matlab