

SNE SIMULATION NOTES EUROPE

SNE Special Issue



Simulation of Technical Systems -
Methods, Tools & Applications

Volume 25 No.2 August 2015

doi: 10.11128/sne.25.2.1029



Journal on Developments and
Trends in Modelling and Simulation

Membership Journal for Simulation
Societies and Groups in EUROSIM

Print ISSN 2305-9974
Online ISSN 2306-0271





EUROSIM 2016

9th EUROSIM Congress on Modelling and Simulation

City of Oulu, Finland, September 12 – 16, 2016



EUROSIM Congresses are the most important modelling and simulation events in Europe. For EUROSIM 2016, we are soliciting original submissions describing novel research and developments in the following (and related) areas of interest: Continuous, discrete (event) and hybrid modelling, simulation, identification and optimization approaches. Two basic contribution motivations are expected: M&S Methods and Technologies and M&S Applications. Contributions from both technical and non-technical areas are welcome.

Congress Topics The EUROSIM 2016 Congress will include invited talks, parallel, special and poster sessions, exhibition and versatile technical and social tours. The Congress topics of interest include, but are not limited to:

Intelligent Systems and Applications
Hybrid and Soft Computing
Data & Semantic Mining
Neural Networks, Fuzzy Systems & Evolutionary Computation
Image, Speech & Signal Processing
Systems Intelligence and Intelligence Systems
Autonomous Systems
Energy and Power Systems
Mining and Metal Industry
Forest Industry
Buildings and Construction
Communication Systems
Circuits, Sensors and Devices
Security Modelling and Simulation

Bioinformatics, Medicine, Pharmacy and Bioengineering
Water and Wastewater Treatment, Sludge Management and Biogas Production
Condition monitoring, Mechatronics and maintenance
Automotive applications
e-Science and e-Systems
Industry, Business, Management, Human Factors and Social Issues
Virtual Reality, Visualization, Computer Art and Games
Internet Modelling, Semantic Web and Ontologies
Computational Finance & Economics

Simulation Methodologies and Tools
Parallel and Distributed Architectures and Systems
Operations Research
Discrete Event Systems
Manufacturing and Workflows
Adaptive Dynamic Programming and Reinforcement Learning
Mobile/Ad hoc wireless networks, mobicast, sensor placement, target tracking
Control of Intelligent Systems
Robotics, Cybernetics, Control Engineering, & Manufacturing
Transport, Logistics, Harbour, Shipping and Marine Simulation

Congress Venue / Social Events The Congress will be held in the City of Oulu, Capital of Northern Scandinavia. The main venue and the exhibition site is the Oulu City Theatre in the city centre. Pre and Post Congress Tours include Arctic Circle, Santa Claus visits and hiking on the unique routes in Oulanka National Park.

Congress Team: The Congress is organised by SIMS - Scandinavian Simulation Society, FinSim - Finnish Simulation Forum, Finnish Society of Automation, and University of Oulu. Esko Juuso EUROSIM President, Erik Dahlquist SIMS President, Kauko Leiviskä EUROSIM 2016 Chair

Info: eurosim2016.automaatioseura.fi, office@automaatioseura.fi

Editorial

Dear Readers – This second SNE issue of the year 2015, the special issue ‘Simulation of Technical Systems – Methods, Tools & Applications’, compiled by the guest editors Thorsten Pawletta (Univ. od Applied Sciences Wismar) and Heinz-Theo Mammen (Hella Corporate Center GmbH, presents selected extended contributions from an ASIM workshop on this subject. Indeed the broad variety of the contributions is very impressive and shows, that modelling and simulation is part of almost all research and development processes for technical systems. The term ‘Technical System’ must be seen in a very general an interdisciplinary context, from design via implementation to operation of a technical systems, as the contributions prove.

The cover page shows the 2nd variation of the algorithmic art design, which Vlatko Ceric, past president of CROSSIM, has provided for the cover pages of SNE Volume 25. Algorithmic art is visual art generated by algorithms that completely describe creation of images. The technique used for the picture series is alienation of ‘classic’ pictures by mainly geometric algorithms – each of the three variations is based on the same classic picture, alienated by the same algorithm, bus with different parameters.

I would like to thank all authors for their contributions, and especially the guest editors of this special issue, who helped to continue the series of SNE special issues, and last but not least thanks to the ARGESIM SNE layout staff for typesetting, preparations for printing, and web programming for electronic publication of this SNE issue.

Felix Breiteneker, SNE Editor-in-Chief, elic@sne-journal.org; felix.breiteneker@tuwien.ac.at

Contents SNE 25(2) - Special Issue ‘Simulation of Technical Systems – Methods, Tools & Applications’

SNE doi: 10.11128/sne.25.2.1029

Editorial Special Issue ‘Simulation of Technical Systems – Methods, Tools & Application’. T. Pawletta, H. Mammen	ii
Potential of Dynamically Adaptable Simulation Models for Virtual Commissionings. P. P. Schmidt, A. Fay	59
MATLAB/Simulink Based Rapid Control Prototyping for Multivendor Robot Applications. . C. Deatcu, B. Freymann, A. Schmidt, T. Pawletta	69
Towards a Newer Toolbox for Computer Aided Polynomial Design of Sampled-Data Systems. R. C. Gomez, B. P. Lampe	79
Simulating a Pneumatics Network using the Modelica Fluid Library. P. Drentel, P. Junglas	85
Efficient Modelling of Heterogeneous Battery Management Systems. T. Markwirth, M. Gulbins, K. Einwich, J. Haase	93
Design and Optimization of an Energy Manager for an Office Building. K. Majetta, C. Clauß, J. Haufe, S. Seidel, T. Blochwitz, E. Liebold, U. Hintzen, V. Klostermann	99
Implementation of Hybrid Systems Described by DEV&DESS in the QSS Based Simulator PowerDEVS. F. Preyser, I. Hafner, M. Rößler	109
Domain-Specific Languages for Flexibly Experimenting with Stochastic Models. D. Peng, T. Warnke, A. M. Uhrmacher ...	117
EUROSIM Societies Info & News	N1-N8

Reader’s Info

Simulation Notes Europe publishes peer reviewed *Technical Notes*, *Short Notes* and *Overview Notes* on developments and trends in modelling and simulation in various areas and in application and theory, with main topics being simulation aspects and interdisciplinarity.

Individual submissions of scientific papers are welcome, as well as post-conference publications of contributions from conferences of EUROSIM societies.

SNE welcomes also special issues, either dedicated to special areas and / or new developments, or on occasion of vents as conferences and workshops with special emphasis.

Furthermore SNE documents the ARGESIM Benchmarks on *Modelling Approaches and Simulation Implementations* with publication of definitions, solutions and discussions (*Benchmark Notes*). Special *Educational Notes* present the use of modelling and simulation in and for education and for e-learning.

SNE is the official membership journal of EUROSIM, the Federation of European Simulation Societies. A News Section in SNE provides information for EUROSIM Simulation Societies and Simulation Groups.

SNE is published in a printed version (Print ISSN 2305-9974) and in an online version (Online ISSN 2306-0271). With Online SNE the publisher ARGESIM follows the Open Access strategy, allowing download of published contributions for free. Since 2012 Online SNE contributions are identified by a DOI (Digital Object Identifier) assigned to the publisher ARGESIM (DOI prefix 10.11128).

Print SNE, high-resolution Online SNE, full SNE Archive, and source codes of the *Benchmark Notes* are available for members of EUROSIM societies.

SNE Print ISSN 2305-9974, SNE Online ISSN 2306-0271

SNE Issue 25(2) August 2015 doi: 10.11128/sne.25.2.1029

→ www.sne-journal.org

✉ office@sne-journal.org, elic@sne-journal.org

✉ SNE Editorial Office, c/o ARGESIM / DWH, Neustiftgasse 57-59, 1070 Vienna, Austria

Editorial SNE Special Issue 'Simulation of Technical Systems – Methods, Tools & Applications'

Modelling and Simulation is an important approach for development and operation of technical systems for many years. Accordingly, the ASIM sections *Simulation of Technical Systems* and *Foundations and Methods in Modelling and Simulation* organize an annual workshop focussed on several topics regarding the simulation of technical systems and general methods and tools in this broad field.

This special issue presents six selected papers from the workshop held in Stralsund, Germany, in June 2015 and locally organized by professor Christine Wahmkow and her team from the University of Applied Sciences in Stralsund. The selected contributions show the wide range of methods and tools used and some considered applications.

The methodological oriented article *Potential of Dynamically Adaptable Simulation Models for Virtual Commissioning* by Puntel Schmidt and Fay investigates the potential of dynamically adaptable simulation models for virtual commissioning of control code deployed on Programmable Logical Controllers.

Deatcu et al. present in the contribution *MATLAB/Simulink Based Rapid Control Prototyping for Multi-vendor Robot Applications* a MATLAB/Simulink toolbox for rapid control prototyping (RCP) of multivendor robot applications. After a general discussion of applying the RCP approach for developing industrial robot controls, the design and usage of an appropriate toolbox is described.

Gomez and Lampe also present a tool development using MATLAB in the third contribution of this issue, *Towards a Newer Toolbox for Computer Aided Polynomial Design of Sampled-Data Systems*. It presents new ideas to update the DIRECTSD toolbox, which realizes recently developed polynomial methods for the analysis and optimal design of sampled-data systems.

The fourth contribution *Simulating a Pneumatics Network using the Modelica Fluid Library* by Drente and Junglas discusses modelling and simulation of a pneumatic network using the Modelica Fluid Library (MFL). It highlights advantages and problems of the MFL to solve such applications.

Markwirth et al. present in *Efficient Modelling of Heterogeneous Battery Management Systems* a simulation environment for the design of battery management systems (BMS) based on the modelling languages SystemC and SystemC-AMS. Moreover, they describe the integration of BMS in COSIDE a design environment for heterogeneous systems.

The sixth contribution *Design and Optimization of an Energy Manager for an Office Building* by Majetta et al. presents objectives and results of the research project enerMAT. A core of enerMAT is the development of Building Energy Management Systems (BEMS) based on simulation and optimization methods.

The contribution by Franz Preyser et al., *Implementation of Hybrid Systems Described by DEV&DESS in the QSS Based Simulator PowerDEVs*, combines two new approaches for implementation of hybrid systems – DEV&DESS with QSS, the Quantized State System. The authors present a method for implementing hybrid models, formulated as DEV&DESS, in the QSS based simulator PowerDEVs.

The last contribution *Domain-Specific Languages for Flexibly Experimenting with Stochastic Models* by Peng et al. discusses and illustrates, how domain specific languages can be used to specify simulation experiments. The authors emphasize on stochastic models, where several problems arise in specifying simulation experiments, such as probability estimation, tolerating stochastic noises, and robustness measurement.

The editors would like to thank all authors, who have contributed to this special issue, and we thank especially for the additional work in improving, extending, and correcting the contributions from the workshop.

Thorsten Pawletta, thorsten.pawletta@hs-wismar.de
University of Applied Sciences Wismar, Germany

Heinz-Theo Mammen, Heinz-Theo.Mammen@hella.com
Hella Corporate Center GmbH, Lippstadt, Germany

Guest Editors SNE Special Issue



ASIM



ASIM



ASIM



ASIM - Buchreihen / ASIM Book Series

Proceedings

- Simulation in Production und Logistics 2015 - 16. ASIM-Fachtagung Simulation in Produktion und Logistik**
M. Raabe, U. Clausen (Hrsg.); ISBN 978-3-8396-0936-1, Stuttgart: Fraunhofer Verlag, 2015. (O. P.)
- Simulation in Produktion und Logistik 2013: Entscheidungsunterstützung von der Planung bis zur Steuerung**
W. Dangelmaier, C. Laroque, A. Klaas (Hrsg.); ISBN 978-3-942647-35-9, HNI-Verlagsschriftenreihe, Heinz Nixdorf Institut, Paderborn, 2013 (O. P.)
- Modellierung, Regelung und Simulation in Automotive und Prozessautomation – Proc. 5. ASIM-Workshop Wismar 2011.** C. Deatcu, P. Dünow, T. Pawletta, S. Pawletta (eds.), ISBN 978-3-901608-36-0, ASIM/ARGESIM, Wien, 2011 (O. A.)
- Simulation in Produktion und Logistik 2010: Integrationsaspekte der Simulation - Technik, Organisation und Personal.** G. Zülch, P. Stock, (Hrsg.), ISBN 978-3-86644-558-1, KIT Scientific Publ. Karlsruhe, 2010 (O.P.)

Monographs

- Simulation und Optimierung in Produktion und Logistik – Praxisorientierter Leitfaden mit Fallbeispielen.**
L. März, W. Krug, O. Rose, G. Weigert, G. (Hrsg.); ISBN 978-3-642-14535-3, Springer, 2011 (O.P.)
- Verifikation und Validierung für die Simulation in Produktion und Logistik - Vorgehensmodelle und Techniken.**
M. Rabe, S. Spieckermann, S. Wenzel (eds.); ISBN: 978-3-540-35281-5, Springer, Berlin, 2008 (O. P.)
- Qualitätskriterien für die Simulation in Produktion und Logistik – Planung und Durchführung von Simulationsstudien.** S. Wenzel, M. Weiß, S. Collisi – Böhmer, H. Pitsch, O. Rose (Hrsg.); ISBN: 978-3-540-35281-5, Springer, Berlin, 2008

Series Fortschrittsberichte Simulation

- P. Einzinger: **A Comparative Analysis of System Dynamics and Agent-Based Modelling for Health Care Reimbursement Systems.**
FBS 25, ASIM/ARGESIM Vienna, 2014; ISBN 978-3-901608-75-9, ARGESIM Report 75 (O. A.)
- M. Bruckner: **Agentenbasierte Simulation von Personenströmen mit unterschiedlichen Charakteristiken**
FBS 24, ASIM/ARGESIM Vienna, 2014; ISBN 978-3-901608-74-2, ARGESIM Report 74 (O. A.)
- S. Emrich: **Deployment of Mathematical Simulation Models for Space Management**
FBS 23, ASIM/ARGESIM Vienna, 2013; ISBN 978-3-901608-73-5, ARGESIM Report 73 (O. A.)
- G. Maletzki: **Rapid Control Prototyping komplexer und flexibler Robotersteuerungen auf Basis des SBC-Ansatzes.** FBS 22, ASIM/ARGESIM Vienna, 2014; ISBN 978-3-901608-72-8, ARGESIM Report 72 (O. A.)
- X. Descovich: **Lattice Boltzmann Modeling and Simulation of Incompressible Flows in Distensible Tubes for Applications in Hemodynamics**
FBS 21, ASIM/ARGESIM Vienna, 2012; ISBN 978-3-901608-71-1, ARGESIM Report 71 (O. A.)
- F. Miksch: **Mathematical Modeling for New Insights into Epidemics by Herd Immunity and Serotype Shift**
FBS 20, ASIM/ARGESIM Vienna, 2012; ISBN 978-3-901608-70-4, ARGESIM Report 70 (O. A.)
- S. Tauböck: **Integration of Agent Based Modelling in DEVS for Utilisation Analysis: The MoreSpace Project at TU Vienna;** FBS 19, ASIM/ARGESIM Vienna, 2012; ISBN 978-3-901608-69-8, ARGESIM Report 69 (O. A.)
- Ch. Steinbrecher: **Ein Beitrag zur prädiktiven Regelung verbrennungsmotorischer Prozesse**
FBS 18, ASIM/ARGESIM Vienna, 2010; ISBN 978-3-901608-68-1, ARGESIM Report 68 (O. A.)
- O. Hagendorf: **Simulation-based Parameter and Structure Optimisation of Discrete Event Systems**
FBS 17, ASIM/ARGESIM Vienna, 2010; ISBN 978-3-901608-67-4, ARGESIM Report 67 (O. A.)

Orders via ASIM (O. A.) or via Publisher (O. P.):

ASIM/ARGESIM Office Germany, Hochschule Wismar, PF 1210, 23952 Wismar, Germany

ASIM/ARGESIM Geschäftsstelle Österreich, c/o DWH, Neustiftgasse 57, 1040 Vienna, Austria

Download (some books) via ASIM webpage in preparation

Info: www.asim-gi.org, info@asim-gi.org

SNE Editorial Board

SNE - Simulation Notes Europe is advised and supervised by an international scientific editorial board. This board is taking care on peer reviewing and handling of *Technical Notes*, *Education Notes*, *Short Notes*, *Software Notes*, *Overview Notes*, and of *Benchmark Notes* (definitions and solutions). At present, the board is increasing (see website):

- David Al-Dabass, david.al-dabass@ntu.ac.uk
Nottingham Trent University, UK
- Felix Breitenecker, Felix.Breitenecker@tuwien.ac.at
Vienna Univ. of Technology, Austria, Editor-in-chief
- Maja Atanasijevic-Kunc, maja.atanasijevic@fe.uni-lj.si
Univ. of Ljubljana, Lab. Modelling & Control, Slovenia
- Aleš Belič, ales.belic@sandoz.com
Sandoz / National Inst. f. Chemistry, Slovenia
- Peter Breedveld, P.C.Breedveld@el.utwente.nl
University of Twente, Netherlands
- Agostino Bruzzone, agostino@itim.unige.it
Università degli Studi di Genova, Italy
- Francois Cellier, fcellier@inf.ethz.ch
ETH Zurich, Switzerland
- Vlatko Čerić, vceric@efzg.hr
Univ. Zagreb, Croatia
- Russell Cheng, rhc@maths.soton.ac.uk
University of Southampton, UK
- Eric Dahlquist, erik.dahlquist@mdh.se, Mälardalen Univ., Sweden
- Horst Ecker, Horst.Ecker@tuwien.ac.at
Vienna Univ. of Technology, Inst. f. Mechanics, Austria
- Vadim Engelson, vadim.engelson@mathcore.com
MathCore Engineering, Linköping, Sweden
- Edmond Hajrizi, ehajrizi@ubt-uni.net
University for Business and Technology, Pristina, Kosovo
- András Jávör, javor@eik.bme.hu,
Budapest Univ. of Technology and Economics, Hungary
- Esko Juuso, esko.juuso@oulu.fi
Univ. Oulu, Dept. Process/Environmental Eng., Finland
- Kaj Juslin, kaj.juslin@vtt.fi
VTT Technical Research Centre of Finland, Finland
- Andreas Körner, andreas.koerner@tuwien.ac.at
Technical Univ. Vienna, E-Learning Dpt., Vienna, Austria
- Francesco Longo, f.longo@unical.it
Univ. of Calabria, Mechanical Department, Italy
- Yuri Merkurjev, merkur@itl.rtu.lv, Riga Technical Univ.
- David Murray-Smith, d.murray-smith@elec.gla.ac.uk
University of Glasgow, Fac. Electrical Engineering, UK
- Gasper Music, gasper.music@fe.uni-lj.si
Univ. of Ljubljana, Fac. Electrical Engineering, Slovenia
- Thorsten Pawletta, pawel@mb.hs-wismar.de
Univ. Wismar, Dept. Comp. Engineering, Wismar, Germany
- Niki Popper, niki.popper@dwh.at
dwh Simulation Services, Vienna, Austria
- Kozeta Sevrani, kozeta.sevrani@unitir.edu.al
Univ. Tirana, Inst.f. Statistics, Albania
- Thomas Schriber, schriber@umich.edu
University of Michigan, Business School, USA
- Yuri Senichenkov, sneyb@dcn.infos.ru
St. Petersburg Technical University, Russia

Siegfried Wassertheurer, Siegfried.Wassertheurer@ait.ac.at
AIT Austrian Inst. of Technology, Vienna, Austria

Sigrid Wenzel, S.Wenzel@uni-kassel.de
Univ. Kassel, Inst. f. Production Technique, Germany

Author's Info

Authors are invited to submit contributions which have not been published and have not being considered for publication elsewhere to the SNE Editorial Office. Furthermore, SNE invites organizers of EUROSIM conferences to submit post-conference publication for the authors of their conferences.

SNE distinguishes different types of contributions (*Notes*):

- *Overview Note* – State-of-the-Art report in a specific area, up to 14 pages, only upon invitation
- *Technical Note* – scientific publication on specific topic in modelling and simulation, 6 – 8 pages
- *Education Note* – modelling and simulation in / for education and e-learning; 6 - 8 pages
- *Short Note* – recent development on specific topic, max. 4 p.
- *Software Note* – specific implementation with scientific analysis, max 4 pages
- *Benchmark Note* – Solution to an ARGESIM Benchmark; commented solution 4 pages, comparative solutions 4-8 pages

Further info and templates (doc, tex) at SNE's website.

SNE Contact & Info

→ www.sne-journal.org

✉ office@sne-journal.org, etc@sne-journal.org

✉ SNE Editorial Office, ARGESIM / dwh Simulation Services,
Neustiftgasse 57-59, 1070 Vienna, Austria

SNE SIMULATION NOTES EUROPE

ISSN SNE Print ISSN 2305-9974, SNE Online ISSN 2306-0271

WEB: → www.sne-journal.org, DOI prefix 10.11128/sne

Scope: Technical Notes, Short Notes and Overview Notes on developments and trends in modelling and simulation in various areas and in application and theory; benchmarks and benchmark documentations of ARGESIM Benchmarks on modelling approaches and simulation implementations; modelling and simulation in and for education, simulation-based e-learning; society information and membership information for EUROSIM members (Federation of European Simulation Societies and Groups).

Editor-in-Chief: Felix Breitenecker, Vienna Univ. of Technology, Inst. f. Analysis and Scientific Computing, Div., Math. Modelling and Simulation, Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria;

✉ Felix.Breitenecker@tuwien.ac.at, ✉ etc@sne-journal.org

Layout / Administration: J. Tanzler, F. Preyser, T. Vobruba; C. Wytrzens, R. Leskovar et al.; Math. Modelling and Simulation Group, Vienna Univ. of Technology, Wiedner Hauptstrasse 8-10, 1040 Vienna, ✉ office@sne-journal.org

Print SNE: Grafisches Zentrum, TU Vienna, Wiedner Hauptstrasse 8-10, 1040, Vienna, Austria

Online SNE: ARGESIM / ASIM, c.o. dwh Simulation Services, Neustiftgasse 57-59, 1070 Vienna, Austria

Publisher: ARGESIM ARBEITSGEMEINSCHAFT SIMULATION NEWS - WORKING COMMITTEE SIMULATION NEWS, Neustiftgasse 57-59, 1070 Vienna, Austria; → www.argesim.org, ✉ info@argesim.org on behalf of ASIM(→ www.asim-gi.org and EUROSIM → www.eurosim.info)

© ARGESIM / EUROSIM / ASIM 2015

Potential of Dynamically Adaptable Simulation Models for Virtual Commissioning

Philipp Puntel Schmidt*, Alexander Fay

Helmut-Schmidt-University, Institute of Automation, Holstenhofweg 85, D-22043 Hamburg

*philipp.puntelschmidt@hsu-hh.de

Simulation Notes Europe SNE 25(2), 2015, 59 - 68
 DOI: 10.11128/sne.25.tn.10291
 Received: August 15, 2015 (Selected ASIM STS 2015
 Postconf. Publ.); Accepted: August 20, 2015;

Abstract. Virtual commissioning (VC) is used to test control code deployed on Programmable Logical Controllers. Simulation models of a plant are the core of any VC approach. Simulation models should represent the plant in a way so that the correct process execution can be tested under customers' conditions. Simulation models of a plant are usually not built monolithically, but by many partial simulation models that represent the modules or components of the investigated plant. To ensure that the VC is efficient and provides helpful results, these partial simulation models can be implemented at different levels of detail, depending on the current test scenario. Usually, the definition of the modules' and components' level of detail is fixed. However, situations exist where more than one level of detail can be adequate. A dynamically adaptable level of detail seems beneficial to e. g. keep computing time at a reasonable level and to ensure meaningful results of the plants simulation model. However, no method or approach exists so far to handle a dynamically adaptable level of detail. The paper presents the research results of the authors on virtual commissioning and focuses on a simulation point of view and is organized as follows: In Section 1, a brief description is given on how to define the right granularity of simulation models used for virtual commissioning. Based on these results, several levels of detail and model types that can be used for a VC approach are introduced in Section 2. In Section 3, situations are described where more than one level of detail is suitable. In Section 4 and Section 5, potentials and challenges of a dynamically adaptable level of detail are discussed and possible solution contributions that could yield benefits for a VC approach are shown.

Introduction

Simulation models that represent the investigated plant are the basis of any virtual commissioning (VC) approach. Usually, these simulation models represent specific elements of a plant like modules and components [1] and are built by many partial simulation models. From a mechatronic point of view [2], these partial simulation models simulate modules that can be seen as noteworthy elements of the overall system and be described as substantial function holders.

Modules realise specific processes or tasks within the plant and bundle their capabilities together to perform the plants greater purpose. Modules themselves are comprised of (hierarchically lower and granular finer) components like sensors, actors and others (Figure 1). Components as typical elements of a module perform - in cooperation and together with an appropriate controller - the specific process/task of the module.

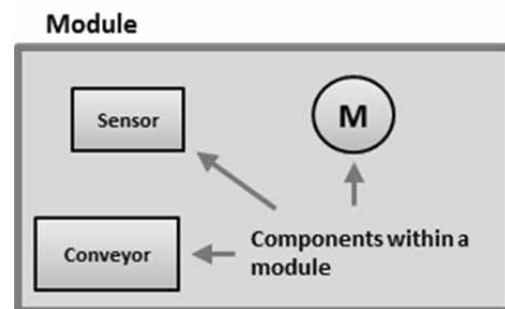


Figure 1: Components within a module.

The granularity of a simulation (what components or modules of a plant are modelled at which level of detail by which model types) takes a high impact on the simulation itself. The more variations of specific elements exist (which widens solution space n), the more difficult it is to realise a simulation model. This applies particularly to an adaptive level of detail, where by principle n elements can be described by m models.

While the number of different model types at different levels of detail and thus the overall number of models increases, kind and quantity of interfaces between these models increase as well (see also Section 2.1). As stated by Haberfellner *et al.* [3], the overall simulation model becomes consequently more detailed by increasing the number of interfaces. Rabe *et al.* [4] refers to Chwif *et al.* [5] and Robinson [6], stating that the intricacy of a simulation model is highly depending on the used level of detail. Eventually, a high and complex level of detail combined with a large amount of possible model types increase the necessary effort on validation of partial parts as well as of the entire model.

As a conclusion, only few specific parts of a manufacturing plant should be simulated at a high and complex level of detail, depending on the specific use case. The largest part of a manufacturing equipment model should be kept as easy as possible, easy to validate. Concerning the level of detail, the following rule should apply: As abstract and less complex as possible, as complex and detailed as needed [7]. Number and kind of specific levels of detail and model types should be restricted.

1 Defining the Right Granularity of Simulation Models

To approach a systematic definition of sensible levels of detail for simulation models used for VC, the appropriate granularity of simulation models should be clarified by means of appropriate model types. Automated plants are structured hierarchically, describing the structure and arrangement of typical elements like sensors, actors and others. Accordingly, a hierarchical perception on plants can be promising. From operator to operator plant hierarchies might be distinguished in detail, which is why a universal, holistic view on plants is not possible without further investigation. Essentially, the well-known systems engineering approach on hierarchy within a system can be determined as a standard [3] when investigating structure and arrangement of automated plants and its involving elements. Furthermore, the data model of the digital plant regarding to VDI4499 [8] should be given one's careful consideration as appropriate database for elements or objects to be identified when looking at a plant's hierarchy and, in the end, architecture.

Therefore, as a prerequisite, the systems' architecture should be processed in a way so that further investigation is possible. According to Systems Engineering standards ([3], [9]) it is common to decompose a system into smaller elements like subsystems, modules and components. Components describe the smallest elements of a plant worth to be considered [28] and are not being split into granularly finer parts. Components represent typical resources like sensors, actors, conveying belts and others. They do not perform processes on their own but in cooperation. Strong interacting components can therefore be described as typical function holders within the automated plants. These function holders in form of (granular more roughly decomposed) modules perform the customers' defined processes based on the interaction of the planned and dimensioned components. Modules can therefore be seen as to be identified parts of the plant that should be individually simulated at an appropriate level of detail. Thus, in preparation to further investigation, it is important to identify modules of a system that can be mapped onto the implemented processes. The first step should be to investigate the systems' architecture.

To identify modules as substantial function holders within the plant, the system architecture DSM, comprising of components and their interrelationships, can be used [10]. The DSM is represented as an n -square (N^2) matrix where components label the rows and columns and are '[...] set in relationship to each other by entering marks or values in the DSM cells' [9]. The relationship can either be binary or, preferably, numerical [10]. Numerical values are advantageous as they allow to weigh the interactions. In this context, Pimmler and Eppinger state that '[...] the number and definitions of the interaction types is dependent upon the context of the given design problem' [10]. The basic procedure for building a system architecture DSM model is described in detail by Eppinger and Browning in [9] and is not further discussed here. Based on the identified interaction of components, several analysis methods on the created DSM model can be used. According to Eppinger and Browning, '[...] the most common method of analysis applied to [system] architecture DSM models is [...] clustering' [9]. Clustering allows for the identification of interacting components in form of clusters that are described as 'a set of components grouped because of certain relationships, suggested through analysis of the [system] architecture DSM, and defined to comprise a module [...]'.

Figure 2 shows this principle: Starting with a set of interacting components, modules can be derived by using the clustering algorithm.

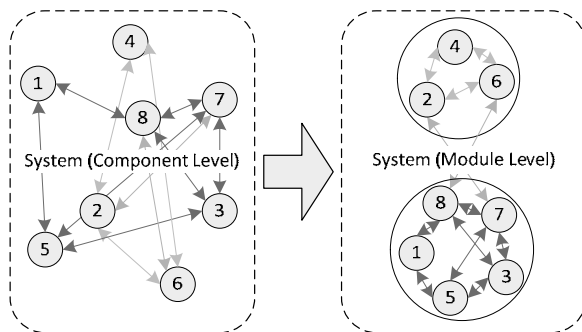


Figure 2: Principle of 'clustering': Components with strong interaction are arranged to clusters (which henceforth form the modules).

By applying the clustering algorithm, modules as substantial function holders can be identified from the system's architecture. Each module usually has numerous and different internal interfaces between the components which it comprises, but only a comparable small number of external interfaces. The latter allow the connection of several modules that altogether represent the investigated plant. The higher the number of interfaces is the more complex a system is [3]. To keep the complexity of a module at a reasonable level, all components within a single module should therefore have the same level of detail in simulation. The different modules, however, will be modelled in appropriate, i.e. possibly different, levels of detail. This will reduce the modelling effort, especially in case of an automated model generation.

2 Model Types and Levels of Detail for Virtual Commissioning

Simulation models of plants' modules and components must be sufficiently meaningful with regard to the respective test case [3]. Models can naturally only be abstractions and simplifications of reality and can therefore only show selected aspects of a system's behaviour. A systematic classification on what aspects should be simulated is beneficial for a VC approach. As defined in [1], four levels of detail (*LoD*) can be identified for a (discrete manufacturing) VC approach and can be described as follows:

- **Macroscopic LoD:** Rough granular perception of the respective plant; modules are represented by a single simulation model (without comprised components). Only time based behavior is considered, goods that are transported and/or processed are not considered.
- **Mesoscopic (not dynamical) LoD:** Fine granular perception of the plant; the module's function is simulated by the interaction of simulation models of components that comprise the module. These components can be devices (like sensors or actors or frequency inverters) or mechanical/pneumatical/hydraulical systems (also called basic system according to [2]). Only time based behavior is considered, goods that are transported and/or processed are considered but movement is restricted to predefined paths (no free movement in space or collisions possible).
- **Mesoscopic (dynamical) LoD:** Fine granular perception of the plant as in the mesoscopic (not dynamical) LoD. Components show physical/dynamical behavior, where applicable. Goods that are transported and/or processed are considered but movement is restricted to predefined paths (collisions on these paths are possible, in contrast to mesoscopic (not dynamical) LoD).
- **Microscopic LoD:** Fine granular perception of the plant as in the mesoscopic (dynamical) LoD. Components as well as transported goods show physical/dynamical behavior. Movement of goods is possible in free space, including any type of collisions like interfering contours, etc.

Several model types (exemplary defined in [1], see also [11]) can be distributed over these four levels of detail and define the solution space used to build the respective plant's simulation model. These model types differ in aspects like time-based and dynamical/physical behavior. The specified model types can be described as follows:

- **Dead time models (macroscopic LoD):** The first model type to be defined can be the simulation of the mechanic and electronic domain in only one model, covering the hierarchical module level. The behaviour of the module is described here as a whole, individual components and therefore concrete domains of a mechatronic system are disregarded. From a system point of view this corresponds to a coarse-granular modelling approach. Models can be seen as black-boxes, they describe a mechatronic system (module)

ignoring its inner structure. By definition the function and behaviour is covered by only one model, this can be done by a simple transformation of the control input to control signals delayed only by a dead time (dead time models). Dead time models on module level can be modelled as simple data flow models since no physical characteristics are considered. Parameterization should be kept flexible, which means that dead time itself could be a parameter or is being calculated e. g. from the transport speed and length of a conveyor. Dead time models are only partly appropriate to describe complex correlations, since models would become very complex and unclear.

Goods or items that are transported or machined are not considered with this model type. Modelling dead time models on module level depend on system knowledge to picture the respective function and behaviour [12]. System knowledge describes the knowledge needed to describe circumstances and processes as a whole. Faults concerning runtime monitoring (e. g. a transport or a process does not finish in time) or discrepancy errors (e. g. errors that describe discrepancies on sensor signals) are to be modelled in an appropriate way and must be suitable to fit deterministic and stochastic investigations.

- Simple device and kinematic models (mesoscopic not dynamical LoD): Consideration of models on module level is often not sufficient, particularly when behaviour and function cannot be modelled in an appropriate way or only with high modelling efforts. A view on the mechanic and electronic domains of a mechatronic system might be necessary. This could be done by considering the component level as identified, where behaviour and function and therefore the individual characteristics of a mechatronic system/module are defined by collaboration of specific components. From a system point of view this corresponds to a fine-granular modelling approach, modelling the behaviour and functions of concrete components. In this particular case, models can be seen as white-boxes, describing the inner structure of a mechatronic system. For the electronic domain, this could be done similarly to module levels models as simple dead time models (named simple device-models). Depending on parameterization, simple device-models show behaviour and function of a device but offer no extra functions or physical behaviour. Simple kinematic models, describing movement and behaviour without moving forces, are the counterpart of simple device

models and represent the model of the basic system. Movement is based on stated paths, whereby movement can be a translation, rotation or combinations of those (defined by the degrees of freedom of the basic system). Simple device- and kinematic models can, as well as dead time models, be modelled as simple data flow models since no physical characteristics are considered. Faults are to be modelled in an appropriate way.

- Complex device and kinematic models (mesoscopic dynamical LoD): If concrete physical behaviour is needed to describe the characteristics of the mechatronic system/module, models of components must provide functions containing equilibrium of power and moments. For the electronic domain, this can be done by models that cover flow and potential of physical systems (as device-models). Simple physical models considering forces that cause movement are the matching models of the basic system. As the previously defined simple kinematic models, movements are limited to stated paths, covering both translational or rotational movements and combinations of those, on this occasion described physically/dynamically. A free movement within space is excluded here. These models are the corresponding partners for device models (named kinetic models), representing the mechanic domain. Faults are to be modelled in an appropriate way. The modelling itself is preferably done by an object-oriented non-causal modelling language suitable for dynamic simulation
- Kinetic models (microscopic LoD): When free movement in space is necessary to cover the characteristic of the mechatronic system/module and its covered process, movements on stated paths are not sufficient anymore. Models must be able to emulate reality in a way that use cases like possible collisions between objects and other structural elements are simulated in a proper way. Models of the basic system therefore depend necessarily on geometric design data [13]. Movement of goods and items is subject to physical rules (complex kinetic models). To simulate physical behaviour of the basic system, models of the device must be simulated in an equivalent level of detail. Device-models as already defined are not suitable, since undefined behaviour of the complex kinetic models could be the consequence when building a simulation model with these models. Device models must be able to react properly to complex kinetic model behaviour, indicating that

complex device-models are needed for the electronic domain. Complex device-models must provide behaviour like switching hysteresis of sensors or s-curves of variable-frequency drives. Functions like the s-curves should be emulated in an appropriate way. As already stated for physical based models, modelling itself is preferably done by an object-oriented, non-causal modelling language, suitable for dynamic simulation. Faults are to be modelled in an appropriate way.

Table 1 shows an overview of the defined levels of detail as well as several important aspects of the particular levels of detail that are important for a VC approach.

Aspect:	Macroscopic	Mesoscopic				Microscopic	
		dev.		mech.		dev.	mech.
		(nd)	(d)	(nd)	(d)		
Consideration of entire process mapping	✓	✓	✓	-	-	✓	-
Time-based behaviour	✓	✓	X	✓	X	X	X
Dynamical/physical behaviour (if applicable)	X	X	✓	X	✓	✓	✓
Failure states	✓	✓	✓	✓	✓	✓	✓
Consideration of processed good*:	X	-	-	✓	✓	-	✓
- Movement of goods on stated paths (= degrees of freedom)	X	-	-	✓**	✓	-	X
- Free movement of goods in space	X	-	-	X	X	-	✓
- Collision of goods possible	X	-	-	X	***	-	✓

dev: device, mech: mechanical /pneumatical/hydraulic basic system (see VDI2206 [2]), nd: not dynamical, d: dynamical

* Good is synonym to bulk goods or piece goods
 ** Acceleration free movement
 *** Only possible in simulated degree of freedom

Table 1: Overview of defined levels of detail and several important aspects of VC.

2.1 Ensuring consistent interfaces between simulation models in different levels of detail

Different levels of detail indicate different modelling techniques. Time based models on a macroscopic resp. mesoscopic (not dynamical) level of detail can be modelled using a causal modelling approach, while simulation models that show dynamical behaviour (mesoscopic dynamical, microscopic level of detail) usually follow a a-causal modelling approach, as already mentioned in the previous section.

However, simulation models in different modelling approaches are not necessarily compatible regarding to their interfaces: a consistent data flow is not possible at all times and must be ensured, especially in case of an automated model generation ([24], [25]) where no manual intervention or correction is demanded. The definition of modelling regulations, e. g. presented by a set of rules, allows for a systematic identification of situations, where additional simulation models like coupling or termination elements (that must be included in the simulation library, see [14] and [27] for an example) are necessary to ensure the interoperability of the partial simulation models. A detailed description on how to solve the problem of inconsistent interfaces is given in [14] and is not further discussed here.

2.2 Deficits when defining the level of detail for certain modules

As described before, each single module can be seen as a substantial function holder within a plant [1]. These function holders perform the customers' defined processes.

Based on available engineering data and heuristics that consider the practical knowledge of the engineers involved, a situation related (test-case specific) required level of detail can be assigned to each single module (see Section 1, also [15]). At the moment, this assignment is fix for the simulation run and cannot change. However, modules often not only perform one process, but many, depending on the character of the appropriate module. Situations can arise where modules can be modelled in multiple levels of detail, according to the appropriate situation and the process that is being executed. A module e. g. executes three different processes where each process requires a different level of detail. An example will be given in the next section.

3 Identifying Situations where an Adaptable Level of Detail is Appropriate

As a first step towards an adaptable level of detail, situations should be systematically identified where the approach is beneficial. An easy transport processes shall be given as example to demonstrate situations where an adaptable level of detail is appropriate.

3.1 Example

A work piece is to be transported over 2 conveyors (CV01, CV03) and a turntable with integrated conveying belt (TT02) there and back again (Process steps P1 to P6 in Figure 3).

All conveying belts consist of one motor; the turntable contains an extra motor for rotation movement. Conveying belt CV01 and CV03 have one sensor each (S1 placed at the beginning of CV01 and S6 placed at the end of CV03) for (accurate) position recognition. Turntable TT02 has 2 sensors (S2 and S3) on its conveying belt. The position of the (rotating) turntable is recognized by two mechanical switches (S4, S5).

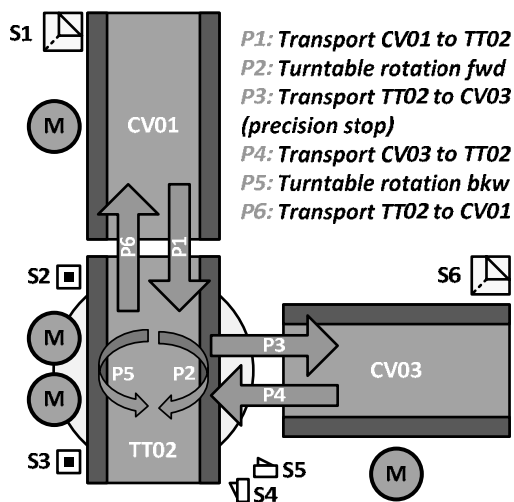


Figure 3: Example as described.

While sensors S2 and S3 can be described as standard capacitive sensors, sensors S1 and S6 allow for precise positioning (yellow shaded, laser distance sensors). Performed processes are P1 to P6 (Figure 3, green arrows).

P1, P2, P4 and P5 can be described as ‘standard’ conveying processes or ‘technical cycles’ that do not have any special requirements e. g. regarding to positioning accuracy. Process P3 and P6 however need very accurate positioning ($\pm 0,1\text{mm}$) for further processing of the work piece (not part of this example model). Figure 4 shows an excerpt of the PLC program that exemplary shows execution of process steps P1 and P2. Eventually, a situation related required level of detail can be defined based on the processes. When performing process P1, P2, P4 and P5, no further requirements on the simulation emerge – the usage of a macroscopic level of detail

(dead time models) seems to be adequate to test the programmed PLC-functions (see also Figure 4).

Following this idea, two levels of detail, depending on the process step, can be suitable for conveyors CV01 and CV03: When P1 and P4 are executed, a macroscopic level of detail (and the included dead-time-model) is sufficient.

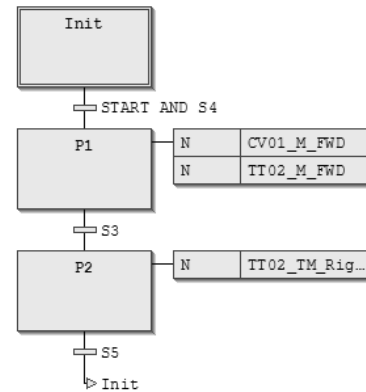


Figure 4: Exemplary PLC program that shows execution of process steps P1 and P2.

However, process P3 and P6 indicate that accurate positioning must be ensured. Parameters like ramps and physical effects like acceleration and friction become important elements of the module models in these process steps, as they influence the quality of the simulation results and, finally, the quality of the PLC-Code itself. The needed physical behaviour can be simulated by a mesoscopic (dynamical) level of detail (complex device and kinematic models, see Chapter 1).

3.2 A systematic way to identify modules that can potentially switch regarding to their level of detail

In case several levels of details are possible, from all suitable levels of detail often the highest and most complex one is chosen to ensure correct behaviour of a module in any case. This can lead to situations where a high and complex level of detail is useless for most of the simulation run since a low (curtailed in the meaning of functionality) level of detail would be sufficient for most of the time. Situations are possible where 1) the computing time increases to a level that prevents VC from running smoothly and 2) where effort on validation can increase to a level where the overall effort/benefit ratio of a simulation is compromised.

An adaptable level of detail where simulation models can switch through different levels of detail (depending on the test case) can be promising. However, there is no known solution for this challenge at the moment. To describe the issue of switching simulation models (regarding to their level of detail) in a standardized and systematic way, the Domain-Mapping-Matrix (DMM) [16] can be used as a supporting tool. The DMM, represented by a rectangular ($n \times m$) matrix, can combine two different domains (in this consideration resources and processes). Rows are represented by appropriate modules (representing the resources of the system investigated) while columns are filled with the process steps that are executed by the modules.

For each module which is involved in a particular process step, the cell at this intersection must be filled with a score. The mapping process is executed by entering a value that represents the strength of the interaction of designed automated processes and processing resources selected during the engineering process itself. In context of the given design problem, each conceivable interaction must be scored.

Depending on the simulation systems' supported levels of detail, the overall quantification scheme of the score could be based on the amount of levels of detail available for modelling. If e. g. four levels of detail (see Section 1) are provided, the scores' values could be based on a range from 1 to 4, in reference to each available level of detail. Therefore, the numbers present the overall needed level of detail (where 1 is defined as macroscopic and 4 as microscopic), ascertained by engineers. Figure 5 shows an example on how the interactions of modules and processes (based on the example in Chapter 2.1) could be weighted.

DMM						
	Process 1	Process 2	Process 3	Process 4	Process 5	Process 6
CV01	1					3
TT02	1	2	1	1	2	1
CV03		!	3	1		

Figure 5: Example of a Domain Mapping Matrix where processes are assigned to modules within a plant. Processes refer to process steps as defined in Chapter 2.1.

Attention should be paid to situations where a module has different scores (exemplarily shown by the red boxes in Figure 5). This can be identified by examining the modules associated row. As a conclusion, the DMM indicates in an easy way to identify situations where more than one level of detail is appropriate. As already described, the highest (and most complex) level of detail of all possible is often chosen for the simulation run to ensure correct behaviour of a module in any case ($\max()$ function over the module's associated row), despite useless for e. g. CV03 when performing process P4 or TT02 performing Processes P1, P3, P4 and P6.

4 Potentials and Challenges of an Adaptable Level of Details

A second step in introducing an adaptable level of detail should focus on possible implementation strategies regarding to the used simulation tool and overall simulation framework and procedure. The main benefit (lower computing time) of an adaptable level of detail should be in focus.

4.1 Possible implementation strategies

Simulation is always strongly depending on the used simulation tool as well as framework and the overall simulation approach like discrete event simulation or continuous simulation. The computing time of a simulation is influenced by multiple factors according to [17]: The type of the used numerical solver (for a detailed comparison of common solvers see [18]), amount of continuous state variables as well as the sheer quantity of events that must be handled by the solver [19]. If an adaptable level of detail is demanded, two very basic implementation strategies can be identified.

1. Implementation in one single tool (e. g. in Modelica [20]): One tool provides all levels of detail in e. g. a (copious) library. The tool must be able to handle discrete event simulation as well as continuous simulation approaches. The tool must be able to swap (partial) simulation models during runtime. This is also called 'integrated simulation' [17].
2. Implementation in different tools according to tool specific abilities and strengths (e. g. Modelica [20] and Matlab/Simulink [21]): Several libraries in different modelling tools provide objects in a specific level of detail. Matlab/Simulink could e. g. be used

for dead-time models (since dead time models have a discrete event character) while a Modelica library can provide models where physical/dynamical behavior is needed.

A middleware (like the Functional-Mockup-Interface (FMI) [22] or implementations with respect to the High-Level Architecture (HLA) [23] standard) is necessary to interconnect these (partial) simulations. The middleware must be able to swap (partial) simulation models during runtime. This principle is also called ‘separated simulation’ according to [17].

Both implementation strategies have their advantages and drawbacks: When implementing all levels of detail feasible in just one single simulation tool, it must be capable to handle both discrete (macroscopic, mesoscopic not dynamical) and continuous (mesoscopic dynamical, microscopic) behaviour (‘Hybrid-Modeling’, compare [17]). Additionally, the simulation tool must provide (numerical) solvers that are capable of handling not only discrete event simulation or continuous simulation, but both. Cardinality of the included simulation library is estimated to be very high, exacerbating servicing as well as model building effort (regardless if models are built by an automatic model approach [24], [25], or manually).

Segregation of discrete and continuous behaviour in different tools (e. g. Matlab/Simulink for discrete event simulation, Modelica for continuous simulation) requires a middleware with standardized interfaces to ensure interoperability of the respective models. Liu and Frey [17] describe this as an intermediate communication and synchronization layer. The well-known Functional Mockup Interface (FMI) [22] is one approach that follows this idea and has already found widespread acceptance in industrial applications.

FMI was introduced with the intention to deliver an interface to develop complex systems where different parts of the system can be modelled in different simulation tools. Simulation models are provided in form of a Functional Mockup Unit (FMU) (exported by the respective simulation tool) and are implemented in the standardized framework. However, FMI increases the complexity of the simulation approach to a level that could make an automatic generation approach according to [24], [25] unfeasible. Efforts on parameterization as well as adjustment of the FMI framework itself are considered to be rather high.

4.2 Influences on computing time when simulating in different levels of detail

Different levels of detail (precisely: different model types) can be indicative of different workloads needed to calculate the plant simulation model, being one of the main reasons an adaptable level of detail might be suitable to avoid unnecessary high computing time. A highly detailed simulation model indicates, due to the high amount of events to be calculated, that computing time may increase to a level that prevents VC from running ‘smoothly’. An example on how smoothness can be defined is provided by [26]: A fixed step of 1ms (hard real-time) of the simulation must be ensured so that the PROFINET communication from simulation model to the PLC is not corrupted within a VC test bench.

As already mentioned, the numbers of triggered events have a large impact on computing time of a simulation model. Each time an event is triggered (e. g. when a sensor triggers or control variables within the PLC change), the deployed solver restarts calculating the whole simulation model. Depending on the solver, these calculations can be iterative (called event-iterations) to ensure the specified simulation accuracy. This increases computing time even further. More complex, very detailed simulation models (that consider physical/dynamical behavior) have naturally a high amount of events and need potentially more computing time than an easy, time based simulation model which is based only on dead-time models (that should only have a low number of events). The defined models types (Chapter 1) should therefore not only differ in the ability to simulate time based and physical/dynamical behavior, but also mainly in the number of events they potentially generate. Furthermore, the used simulation models should be highly optimized to a specific solver (e. g. in [26], the Euler solver is used) to ensure a hard real-time at any time whenever needed.

4.3 Conclusions and thoughts about an adaptable level of detail

As a conclusion, the used simulation models, e. g. stored in a model library (see [27] for further details and an example), should be optimized in a way so they generate as less events as possible. Additionally, it must be ensured that the number of events grows with the respective level of detail and not vice versa: models at a macroscopic level of detail should generate (clearly) less events than models at a microscopic level of detail.

This highly correlates with the solver used and must be taken into consideration when building or optimizing a simulation library that contains models at different levels of details. The main benefit and the ultimate goal of an adaptable level of detail, is therefore mainly depending on the events generated and the solver used, not necessarily from the simulation principle itself (discrete event/continuous simulation). Additionally, the interoperability of the simulation models must be ensured at all time, not only when the level of detail is determined static, but especially when simulation models should change during runtime.

On the technical side, no possibility to change models during runtime exists, to the authors' best knowledge, so far, and this holds for both implementation possibilities (integrated/separated simulation). This can be seen as a major point that should be taken into consideration in future investigations.

5 Requirements for Implementing an Adaptable Level of Detail

As concluded in Chapter 3, the used simulation tool, simulation framework and number of events have a huge impact on the implementation of an adaptable level of detail. Several requirements can be identified that must be accomplished regardless of the used simulation environment. These requirements represent important factors that not only concentrate on the simulation model or the used framework, but also on the simulation (building and executing) process itself and shall be considered when an adaptable level of detail is to be implemented in future applications:

- It must be possible to generate the simulation model automatically: Information potentially necessary for a dynamical switching should not prevent an automatic generation approach (e. g. described by [24] or [25]).
- The level of detail may only be switched considering modules: Only complete modules (including all comprised components) may be switched through different levels of detail, not the components comprising a module (the goal is to keep complexity of the simulation model as well as intricacy of the switching process itself on a reasonable and manageable level).
- Interoperability must be ensured: Interfaces between partial simulation models should always be compatible, even when these models are switched between several levels of detail.

- Computing time shall not increase while 'switching' the simulation models: Effort on 'switching' simulation models to different levels of detail should be kept to a minimum regarding to computing time to ensure real time ability.
- 'Switching' simulation models within a process step must be prevented: A specific level of detail assigned to a module must be assigned fix until the process is terminated. Switching levels of detail is therefore not allowed within a process, but between two subsequent processes.
- It must be clearly identifiable what process is being executed: In case a module is able to execute several processes, the simulation tool/framework must be capable to identify the specific process.
- Models used for VC should always be optimized to generate the minimum possible number of events: The model library (an example can be found in [27]) must be optimized and it must be ensured that simulation models in a low level of detail create significant fewer events than models at a high level of detail. This can mean that models should be optimized to run with a specific solver in a specific simulation environment.

6 Conclusion

The paper presents considerations regarding an adaptable level of detail when running a simulation with the purpose of virtual commissioning. Furthermore, an outlook on how an adaptable level of detail might be implemented according to simulation library and simulation framework was given. While different levels of detail have already been implemented in several test cases [1], at the moment no known simulation tool as well as simulation framework is capable of switching simulation models during runtime of the simulation. Future work on the topic should be focused on two aspects: Firstly, a refined method to identify situations where an adaptable level of detail is suitable (based on the method introduced in this paper) and secondly a technical solution that enables swapping simulation models at different levels of detail.

References

- [1] Puntel Schmidt P, Fay A. Levels of Detail and Appropriate Model Types for Virtual Commissioning in Manufacturing Engineering. In: *Proceedings MATHMOD 15 – Vienna Conference on Mathematical Modelling, Vienna, 17.-20. February 2015.*

- [2] VDI2206: *Design methodology for mechatronic systems*. VDI-Verlag, Düsseldorf, 2004.
- [3] Haberfellner R, de Weck O, Fricke E, Vössner S. *Systems Engineering. Grundlagen und Anwendungen*. Orell Füssli Verlag AG, Zürich, 2012.
- [4] Rabe M, Spieckermann S, Wenzel S. Verification and Validation for Simulation in Production and Logistics. *Simulation News Europe*. 2010; 19(2):21-29.
- [5] Chwif L, Pereira B MR, Paul RJ. On simulation model complexity. In: Joines JA, Barton RR, Kang K, Fishwick PA, *Proceedings of the 2000 Winter Simulation Conference*, Orlando (USA), IEEE, Piscataway, 2000. P. 449-455.
- [6] Robinson S. *Simulation: The Practice of model development and use*. John Wiley & Sons, Chichester, 2004.
- [7] Hrdliczka V. Leitfaden für Simulationsbenutzer in Produktion und Logistik. *ASIM Mitteilungen* 58, 1997.
- [8] VDI4499: *Digital Factory*. VDI-Verlag, Düsseldorf, 2008.
- [9] Eppinger SD, Browning TR. *Design structure matrix methods and applications*. Cambridge, Mass: MIT Press (Engineering systems), 2012.
- [10] Pimpler TU, Eppinger SD. *Integration analysis of product decompositions*. Cambridge, Mass: Alfred P. Sloan School of Management, Massachusetts Institute of Technology (Working paper / Alfred P. Sloan School of Management, WP # 3690-94-MS), 1994.
- [11] VDI3693: *Virtuelle Inbetriebnahme. VDI-Richtlinien-Entwurf*. VDI-Verlag, Düsseldorf, 2015.
- [12] Lüdecke A, Pelz G. Top-Down Design of a Mechatronic System. In: *Third Forum on Design Languages (FDL 2000)*, P. 151-158. Tübingen, Germany, 2000.
- [13] Reinhart G, Lacour FF. Physically based Virtual Commissioning of Material Flow Intensive Manufacturing Plants. In: *3rd International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV 2009)*. Munich, Utz, 2009.
- [14] Puntel Schmidt P, Fay A. Konsistente Simulationsmodelle für die virtuelle Inbetriebnahme fertigungstechnischer Anlagen mit Hilfe regelbasierter Modellverbinder. In: Rabe M, Clausen U, editors. *Fraunhofer IRB Verlag, Stuttgart 2015 – Tagungsband der 16. ASIM Fachtagung Simulation in Produktion und Logistik. Simulation in Production and Logistics 2015*, Dortmund, 23.-25. September 2015.
- [15] Puntel Schmidt P, Fay A. Applying the Domain-Mapping-Matrix to Identify the Appropriate Level of Detail of Simulation Models for Virtual Commissioning. In: *2nd IFAC Conference on Embedded Systems, Computational Intelligence and Telematics in Control Schedule (CESCIT)*, Maribor, Slovenia, 2015.
- [16] Pimpler TU, Eppinger SD. *Integration analysis of product decompositions*. Cambridge, Mass: Alfred P. Sloan School of Management, Massachusetts Institute of Technology (Working paper / Alfred P. Sloan School of Management, WP # 3690-94-MS). 1994.
- [17] Liu L, Frey G. Efficient Simulation of Hybrid Control Systems in Modelica/Dymola. *Proceedings of the 6th Vienna International Conference on Mathematical Modelling (MATHMOD 2009)*, Vienna, Austria, Feb. 2009. pp. 1344-1352.
- [18] Liu L, Felgner F, Frey G. Comparison of 4 Numerical Solvers for Stiff and Hybrid Systems Simulation. *Proceedings of the 15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2010)*, Bilbao, Spain, Sept. 2010.
- [19] Liu L, Felgner F, Frey G. Modellierung und Simulation von Cyber-Physical Systems, *Proceedings of the 12th Fachtagung Entwurfkomplexer Automatisierungssysteme (EKA 2012)*, Magdeburg, Germany, May 2012. pp. 149-157.
- [20] Modelica Association [Internet]. [cited 2015 May 05] Available from: www.modelica.org
- [21] MATLAB/Simulink v8.4. Natick, Massachusetts: The MathWorks Inc., 2014.
- [22] Modelisar, FMI for Model Exchange 1.0 Specification [Internet]. [cited 2010]. Available from: https://svn.modelica.org/fmi/branches/public/specifications/FMI_for_ModelExchange_v1.0.pdf
- [23] IEEE 1516: *Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules*. 2010.
- [24] Puntel Schmidt P, Fay A, Riediger W, Schulte T, Köslin F, Diehl S. Validierung von Steuerungscode mit Hilfe automatisch generierter Simulationsmodelle. In: *at – Automatisierungstechnik*. 2015; 63(2):111–120.
- [25] Barth M, Fay A. Automated generation of simulation models for control code tests. In: *Control Engineering Practice*. 2013; 21(2):218-230.
- [26] Riediger W, Puntel Schmidt P, Köslin F, Schulte T. Hardware-in-the-Loop-Simulation fertigungstechnischer Anlagen. In: *SPS/IPC/DRIVES 2014*, Nürnberg, Germany, 25. -27. Nov. 2014.
- [27] Köslin F, Puntel Schmidt P, Riediger W, Fay A. Entwurf einer Modelica Simulationsbibliothek für die virtuelle Inbetriebnahme fertigungstechnischer Anlagen. In: *7th International Symposium on Automatic Control, AUTSYM 2014*, Wismar, Germany, 25.–26.09.2014.
- [28] Göring M, Fay A. Automation Systems – Formal Modeling of Temporal Change of Physical Structure. In: *Proceedings of the 'IEEE IECON 2012, the 38th Annual Conference of the IEEE Industrial Electronics Society'*, Montréal, Canada, 2012.

MATLAB/Simulink Based Rapid Control Prototyping for Multivendor Robot Applications

Christina Deatcu^{*}, Birger Freymann, Artur Schmidt, Thorsten Pawletta

Hochschule Wismar – University of Applied Sciences: Technology, Business and Design,
Research Group Computational Engineering and Automation, Philipp-Müller-Straße 14, 23966 Wismar, Germany
^{*}*christina.deatcu@hs-wismar.de*

Simulation Notes Europe SNE 25(2), 2015, 69 - 78
DOI: 10.11128/sne.25.tn.10293
Received: July 10, 2015 (Selected ASIM STS 2015
Postconf. Publ.); Accepted: July 20, 2015;

Abstract. Industrial robots are used in various fields of application and many robot manufacturers are active in the market. In most cases, their software solutions are proprietary and, consequently, they cannot be used for third party robots. Moreover, the integration of external hard- or software is highly restricted. Long term standardization efforts for robot programming languages, such as the Industrial Robot Language (IRL) and its successor, the Programming Language for Robots (PLR), have been mostly ignored by robot manufacturers. This fact leads to a restriction on the combined usage of robots. Multi-robot applications where robots have to interact are usually limited to software solutions and robots of one manufacturer. On the other hand, control design in engineering is often carried out by the usage of Scientific and Technical Computing Environments (SCEs) like MATLAB. The Robotic Control & Visualization Toolbox (RCV Tbx) for MATLAB/Simulink tries to close the gap between robot manufacturer-specific software solutions and SCEs. The current version of the RCV Tbx supports a uniform and integrated control development for KUKA and KAWASAKI robots in the MATLAB/Simulink environment. An extension to other robot types is straight forward. Thus, the implementation of heterogeneous multi-robot applications is considerably simplified.

Introduction

This paper is an extended version of [1] and aims to introduce the RCV Tbx for MATLAB/Simulink as an easy to use Rapid Control Prototyping (RCP) Tool for multivendor robot controls. The RCV Tbx has been under development by the research group *Computational Engineering and Automation* (CEA) at Wismar University since 2009 [2].

As research into robotics is proceeding rapidly and new fields of application for robots are being made up continually, the requirements concerning robot control development are increasing, too. Fast and easy control programming, integration of external hardware or software components and multi-robot operation are of particular importance. In this context it is often desirable to use a SCE, such as MATLAB, for RCP. RCP, according to Abel and Bolling [3], is understood as an integrated, continuous control development from early design to operating phase in a homogenous environment.

In addition to multi-robot capability, multivendor applications are one further key aspect. Today, various robot manufacturers are established on the market. They offer proprietary software environments with special robot programming languages such as KRL (VEN KUKA Robotics), AS (VEN Kawasaki Robotics) or RAPID (VEN ABB Robotics). From a software engineering point of view, all these robot languages are pretty similar. Nevertheless, long-term standardization efforts like the IRL and its successor, the PLR are still unsuccessful. In addition, almost all robot manufacturers offer a Computer Aided Robotic (CAR) system, which is also referred to as 3D robot simulation software. CAR systems typically provide physics-based robot models for one manufacturer as well as interfaces to 3D CAD systems. Thus, a simulation and 3D visualization of complete robot cells is supported. Robot controls can be developed within these virtual environments using the proprietary robot languages. Such CAR systems simplify the development and commissioning of robot applications. However, the proprietary software limits applications to products from its manufacturer. There are some third-party CAR systems available, such as 3DRealize-R by Visual Components Corporation [4]. They offer a comprehensive solution for simulation and 3D visualization of heterogeneous robot types.

However, the control programming is still based on the different proprietary robot languages. Hence, the development of interacting multi-robot controls is complicated for robots from different manufacturers.

For MATLAB/Simulink, besides the RCV Tbx [2], there exists a robot control toolbox for KUKA robots (KUKA Control Toolbox, KCT) developed at the University of Sienna [5, 6]. The KCT connects a remote MATLAB computer via TCP/IP to the robot controller of a KUKA robot. Usage of KCT is limited to one single robot from one manufacturer, namely KUKA, so that multi-robot and multivendor applications cannot be addressed.

Inspired by the idea of RCP in control theory, the research group CEA began in 2004 to develop a MATLAB KRL toolbox. This toolbox supports control programming of KUKA robots within MATLAB including the usage of all available MATLAB features. Moreover, it provides a first MATLAB based CAR system [7]. As well as the KCT, the MATLAB KRL Tbx was limited to KUKA robots, but users already benefited from the powerful methods as well as the various interfaces provided by MATLAB and its toolboxes.

Almost all of the proprietary robot languages are imperative languages containing similar programming elements. Thus, the approach of the MATLAB KRL toolbox was generalized. A uniform robot control language for robot types from different manufacturers has been developed with the RCV Tbx for MATLAB. Moreover, the simulation and 3D visualization tools have been enhanced.

This paper is organized as follows: Section 1 introduces the concept of RCP and relates it to robot controls. In Section 2, the RCV Tbx for MATLAB is described. Design, implementation as well as user interface aspects are analyzed. Finally, Section 3 gives a summary and identifies potentials for future work.

1 Rapid Control Prototyping

This section summarizes the RCP approach and how it can be used for robot control development. Systematic development of controls can be carried out following the V-model derived from Orth, Abel and Bollig [8, 3].

The V-model defines two main phases, the design phase and the commissioning phase. The design phase starts with the problem specification and continues with a draft and simulative testing of the control algorithms.

It is completed with coding of an executable control program for the target hardware. This piece of software is then used to bring the control into operation. The commissioning phase starts with component tests and ends with a test of the control across the entire process. Each step of the V-model may have to be performed several times and through several iterations or it is possible that leaps back in the development process will occur.

1.1 Fundamentals

RCP generally requires either a well-adjusted tool chain to follow the V from specification phase down to coding phase and up to operational phase or support from an integrated development environment. This integrated development environment can be an SCE. Furthermore, Software-in-the-Loop simulation (SiL) and Hardware-in-the-Loop Simulation (HiL) techniques following Abel and Bollig [3] and explicit automatic code generation are key features of RCP-capable software systems.

Maletzki [9] adopted the general V-model for controls for robot control development as illustrated in Figure 1.

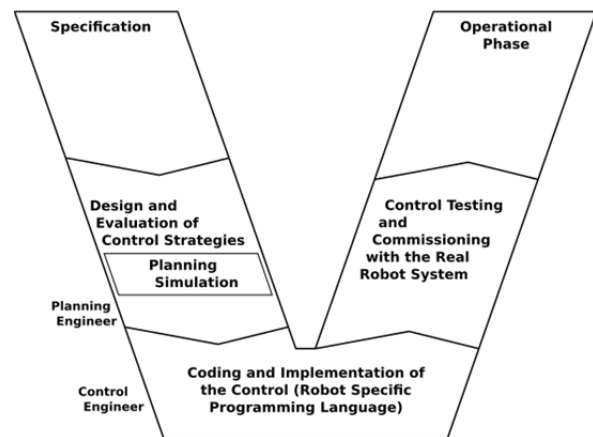


Figure 1: Adopted V-Model for Robot Control Development According to Maletzki [9].

A notably critical point is the transfer of results from planning simulation to the coding and implementation phases. As for industrial robots, the executable control program for the target hardware typically has to be written in a robot specific programming language; a continuous tool chain is not guaranteed. The control strategies that result from the planning simulation are handed over from the planning engineer to the control engineer. This kind of manual handing over is obviously fault-prone.

1.2 RCP and proprietary robot controls

Conventionally, robot controls are coded in vendor-specific programming languages and tested using dedicated 3D-simulation software. Such CAR systems offer software libraries with robot models from the particular vendor and usually include interfaces to 3D-CAD software. In CAR systems, concrete control strategies can be implemented within the specific robot programming language so that extra coding after simulative testing of the control is not required. 3D visualizations, of e.g. robot movements and potential collisions, play a decisive role if CAR systems are deployed. Continuity as required for RCP is partly given but manual knowledge transfer from planning to control engineer is still necessary.

The conventional robot control programming approach supported by CAR systems and associated languages and software respectively are depicted in Figure 2.

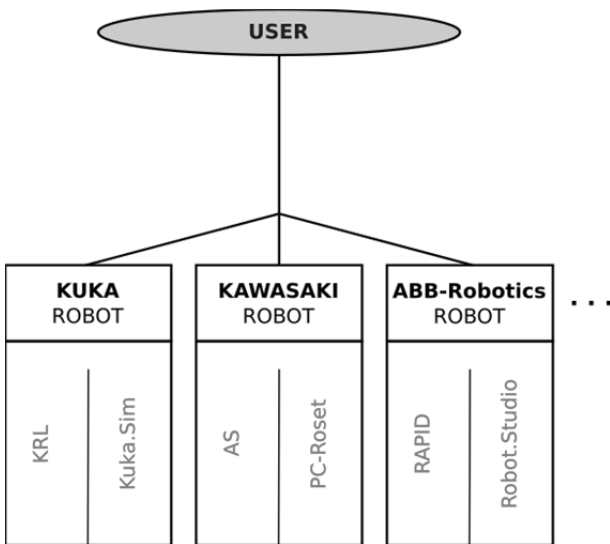


Figure 2: Conventional robot control development using vendor specific software.

After specification of the control requirements the planning engineer designs control strategies using planning simulation tools.

Subsequently, he delivers a control strategy to a control engineer mostly in a textual manner. The control engineer implements the robot control using vendor specific tools as listed in Figure 2. At this point there is a break in the tool chain according to the definition of RCP by Abel et al. [3].

However, a continuous reuse of software components during this transition is hard to realize for robot control development, because different concepts are used and are necessary in these two phases. Starting from this point, a control engineer can implement and deploy a robot control for vendor specific robots compliant with the definition of RCP. For example, for KUKA robots, coding of controls is done using KUKA Robot Language (KRL) and simulation of the control with robot models is carried out using the Kuka.Sim software. Control testing and commissioning with the real robot system can be achieved by automatic code generation or a communication link as illustrated in Figure 3.

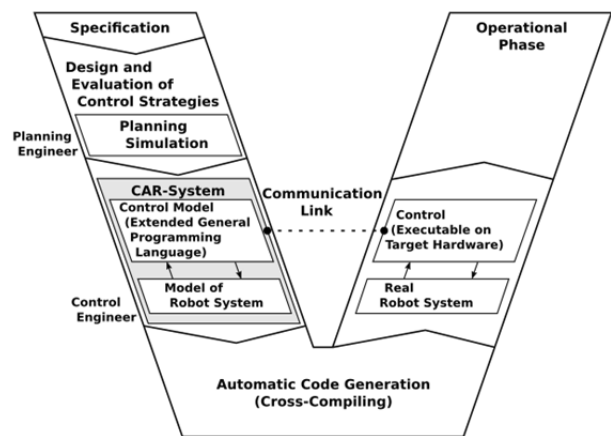


Figure 3: Detailed V-Model for robot control development according to Maletzki [9].

In the case of industrial robots, an explicit code generation for target hardware is not necessary. The control program from the design phase is implemented in a robot-oriented language and running on a robot control computer that is used in operation phase. In the context of RCP, this approach is called *implicit code generation*. Hence, commissioning can be carried out via a simple communication link between a CAR system on a PC and a robot controller. This practice is called *Software in the Loop* (SiL) approach in [3]. The concept of explicit automatic code generation for target hardware is important for mobile robotic applications.

However, the robot control development software depicted in Figure 2 is vendor-specific and incompatible between types. This fact complicates the previously discussed control development or makes it nearly impossible to develop multivendor robot applications.

1.3 RCP and the RCV Toolbox for MATLAB/Simulink

Controls for interacting robots and multivendor robot controls are examples of advanced tasks in robot control development.

Figure 4 illustrates how the RCV Tbx eases such tasks by providing not just a generalized interface to robot control and robot visualization commands, but also to robots from different vendors and/or types. The control engineer implements a robot control as a MATLAB coded sequence of control commands. These commands are independent from a specific robot vendor as well as from a specific robot language, but they are very similar to established robot programming languages such as KRL and AS. Furthermore, control commands can be combined with any MATLAB commands and can also be generated from Stateflow or Simulink models. This option eases the implementation of complex controls by making available high level programming concepts.

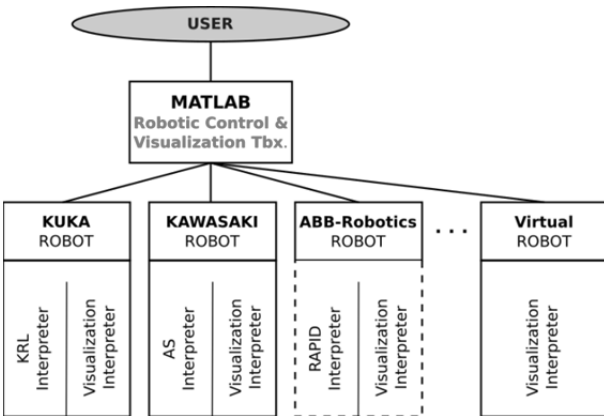


Figure 4: Robot control programming with the RCV Tbx for MATLAB/Simulink.

Moreover, the RCV Tbx for MATLAB provides robot-specific interpreters coded by using the vendor-specific robot languages, which have to be installed on the robot controllers.

The MATLAB based control PC and the robot controllers are connected via a bidirectional communication link. The interpreters are responsible for the identification and execution of control commands that are transmitted by the MATLAB based control PC and they deliver acknowledgements or sensor signals back. Figure 5 illustrates an RCV Tbx-based multi-robot configuration.

One or more computers with MATLAB/Simulink and the RCV Tbx installed act as the continuous software environment from the early design and evaluation phase via control testing with the real system and finally the operational phase. No recoding or reimplementa-tion is necessary, rather the control program can be extended successively until it meets the de-mands of the intended control task. SiL and HiL approaches as defined in [3], as requirements for RCP capability of control development, are met because the simulated control can be stepwise extended to become the real control for the operational phase.

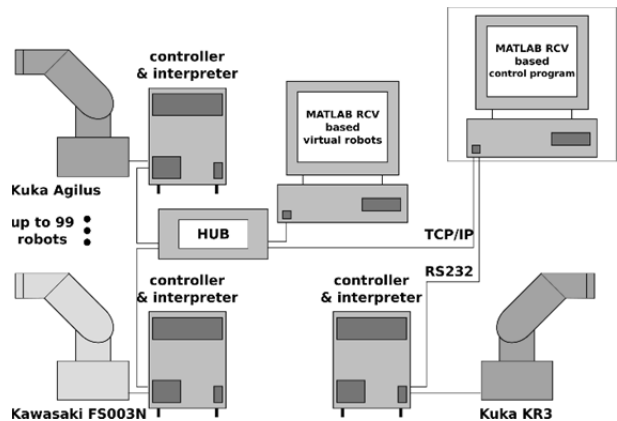


Figure 5: A Multi-robot configuration with heterogeneous robot types using the RCV Tbx.

Short control sequences or single control tasks can already be tested with the real process during design and evaluation (SiL). HiL tests, in which a real control device is tested with a simulated process, are not usually relevant for industrial robot controls as mentioned in Section 1.3. However, they could be of interest for other devices in a robot application. Furthermore, a control application can also be tested in a purely virtual environment using the RCV Tbx similar to the usage of a CAR system. For this purpose the RCV Tbx provides 3D-models of real robots and other virtual components.

2 Design and Usage of RCV Tbx for MATLAB/Simulink

This section describes and analyzes the software design of the RCV Tbx. It details the two toolbox downloads as they are available at [2], namely the robot control download (*Robotic Control Tbx*) and the visualization download (*Robotic Visualization Tbx*).

It is shown how a robot control program can be developed, tested by simulation, put into service (soft commissioning) and finally used as a real control. Continuity of the tool chain, as is needed to meet the requirements of RCP, is given. Hence, the same sequence of control commands coded in the same language can be sent either to a visualized or a real robot.

Figure 6 depicts the main functional parts of the RCV Tbx. *Control*, *Interpreter* and *Visualisation* are the three main functionalities which are distributed across the two software packages.

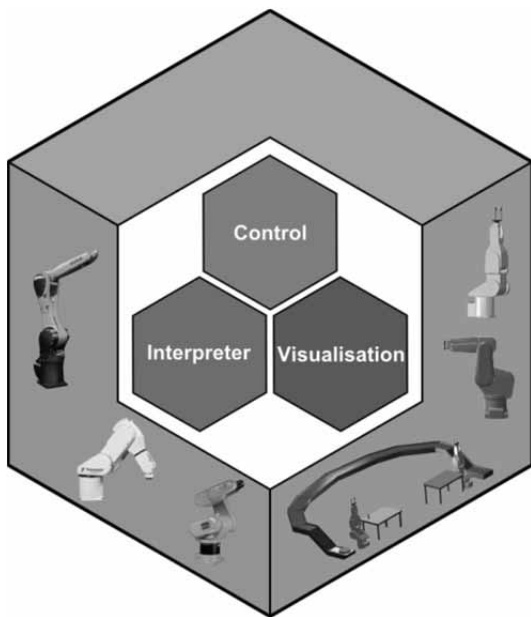


Figure 6: Main parts of the RCV toolbox for MATLAB/Simulink.

A collection of control commands for real, as well as visualized, robots and interpreters for real robots are downloaded with the first part of the RCV Tbx, the *Robotic Control Tbx*. Interpreters for visualized robots, also called virtual robots, and the visualization software are downloaded with the second part of the RCV Tbx, the *Robotic Visualization Tbx*. Currently, both parts support interpreters for Kawasaki FS003N, Kuka Agilus KR6 (TCP/IP connection) and for KUKA KR3 (RS232 connection).

Interaction between the MATLAB control PC and the interpreter program as well as the interpreter algorithm is depicted in Figure 7.

The interpreter can be installed either on a robot controller or on a MATLAB PC as part of the *Robotic Visualization Tbx*. Implementation of the interpreter slightly differs depending on robot type and robot controller hardware. However, the basic principle of interpreter program algorithm remains the same.

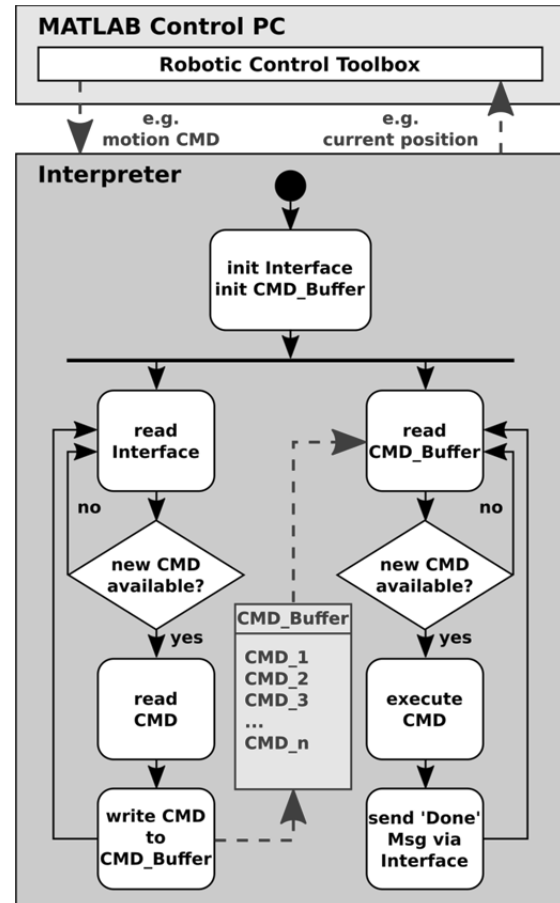


Figure 7: Simplified interpreter algorithm.

The connection between interpreter and MATLAB control PC is always established by the interpreter. It can either be an RS232 or a TCP/IP link. After connection is established and communication interface as well as the command buffer is initialized, robot control commands can be received by the interpreter. Commands are written to a FIFO buffer, read from there and brought to execution. Buffering the commands is necessary, because execution of most commands takes some time.

The interpreter program is not blocking meanwhile and receiving of subsequent commands is always possible. Some commands, such as commands for stopping the robot for security reasons or altering the motion speed during an ongoing robot movement, have to be executed immediately. Handling of those commands is not shown in Figure 7.

2.1 Part I: Robotic Control Tbx

This subsection details robot control programming and gives some examples of robot control commands.

The control of real robots requires an interpreter program to be installed on the robot controller. The interpreter for each kind of real robot is written in the appropriate robot-specific language and copied to the robot controller. The interpreter program takes the commands from MATLAB and translates them into commands for the robot-specific language which are executable by the robot controller. Security considerations, such as workspace supervision and movement execution, are, therefore, still covered by the robot controller.

After the toolbox is installed on the control PC and the appropriate interpreter is copied to the robot controller, the interpreter program on the controller needs to be started. For RS232 connections, after startup of interpreter, the controller is ready to receive MATLAB control commands immediately. For TCP/IP connections, the interpreter on the controller acts as a server and waits for a suitable client to connect. This client is a robot object created on the control PC.

Table 1 lists all available commands for robot control alphabetically.

Of essential importance is the `robot()` command which creates and destroys the robot object which acts as an interface for control commands. Listing 1 shows the syntax of the `robot()` command. Currently, robots can either be of type 'Kawasaki' or of type 'Kuka'.

```
>> h1 = robot( 'open', 'Kawasaki', ...
              'tcpip', IP-ADDRESS, PORT );
>> h2 = robot( 'open', 'Kuka', ...
              'serial', COM-PORT );
>> robot( 'close', h1, h2 );
```

Listing 1: Create and destroy robot objects.

<code>rbrake()</code>	brake the motion of one or all robots directly
<code>rcallback()</code>	define functions, that are executed automatically
<code>rdisp()</code>	formatted display of position structures
<code>rerror()</code>	set up a function, dealing with error codes received from robot controllers
<code>rget()</code>	get interpreter-, toolbox- and motion-properties and current positions
<code>ris()</code>	check the status of commands and processes
<code>rkill()</code>	brake and stop the motion of one or all robots directly; then set up the original state of robot(s)
<code>rmove()</code>	move a robot
<code>robot()</code>	create or destroy a robot object that can be controlled
<code>rpoint()</code>	define robot-positions with coordinates and motion-properties
<code>rprocess()</code>	define complex operations
<code>rreset()</code>	reactivate robot(s) after automatic switching-off of the interpreter(s)
<code>rrun()</code>	reactivate robot(s) after using <code>rbrake()</code> or <code>rstop()</code>
<code>rset()</code>	set up interpreter-, toolbox- and motion-properties
<code>rstatus()</code>	switch ON or OFF status report
<code>rstop()</code>	brake and stop the motion of one or all robots directly
<code>rteach()</code>	teach and save positions of a robot
<code>rwait()</code>	block the MATLAB-prompt until the status of commands or processes fulfills a condition

Table 1: List of available robot control commands.

The command `robot()` creates a robot object and returns a MATLAB handle to the object. This handle can then be passed to other control commands to address this specific robot. Some control commands such as `rbrake()`, `rkill()`, `rreset()`, `rrun()` and `rstop()` affect all robots that the control PC is currently connected to, if no handle is passed to the command as a first parameter. Complex control operations can be coded by using the command `rprocess()`. With this command, tasks to be fulfilled by more than one robot can also be defined. Sequential as well as concurrent operations are possible, and control commands can be structured and grouped. Listing 2 shows a short example for a concurrent operation with two robots.

```

>> load P1 P2
>> rprocess( { ...
    Kuka, P1, ...
    Kawasaki, {P2, 'speed', 50}, ...
}, ...
{
    Kuka, 'home', ...
    Kawasaki, 'home', ...
} );

```

Listing 2: Two robots in a concurrent operation programmed with `rprocess()`.

`Kuka` and `Kawasaki` are the handles for the robots. The robots move at the same time to the positions `P1` and `P2` which are defined by the loaded variables `P1` and `P2`. They start to move to their home positions only after they have both finished their moves.

The `rprocess()` command already allows some complexity, but furthermore, all standard and advanced programming features of MATLAB/Simulink can be used to create even more complex and also simulation based robot controls. An example of such integration with other tools is depicted in Figure 8 where RCV control commands are embedded in Stateflow.

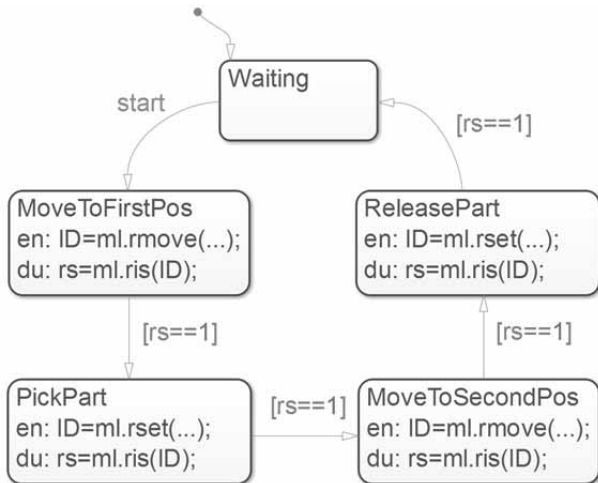


Figure 8: State-based control using Stateflow and RCV Tbx.

The integration of arbitrary external hardware, which can act as sensors or actors, is also feasible. This would allow a big advantage compared to the restricted availability of additional hardware when developing robot controls with proprietary software and languages.

2.2 Part II: Robotic Visualization Tbx

The *Robotic Visualization Tbx* is the second software package of the RCV Tbx and can be used for testing and enhancement of controls developed with the Robotic Control Tbx. It offers distributed 3D-visualization of up to 99 robots in MATLAB and allows interactive control of visualized real robots and also pure virtual robots in a virtual environment. Thereby, safe development, testing and debugging of robot applications is possible. Furthermore, the toolbox includes an STL interface for importing user-defined graphical objects designed using external CAD software.

The control of a visualized robot requires an interpreter program, too. Interpreter functionality for virtual robots is included in the *Robotic Visualization Tbx*. Interpreters are part of virtual robot objects and establish a TCP/IP link between control PC and visualization PC. Control program and visualization can physically be located on the same PC, represented by two MATLAB instances. User toolbox's interface as well as virtual objects is designed as a MATLAB class. Figure 9 pictures visualizable object types that map entities of the real world.

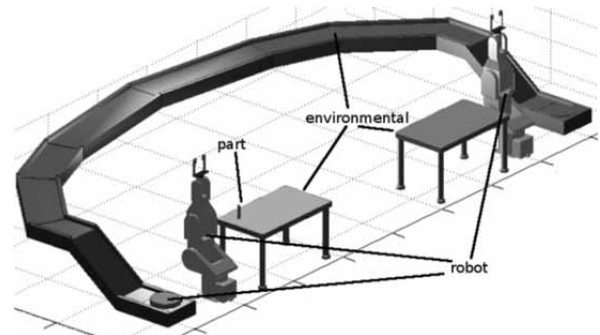


Figure 9: Types of visualizable object.

Besides virtual robot objects environmental objects and part objects can be visualized. Environmental objects are passive objects that cannot be moved by the robots. Part objects are also passive objects, but they can be picked and moved by robots. In Figure 9 examples for environmental objects are two tables and a conveyor; a test tube represents the part objects. Robot objects that represent the active visualization objects include kinematics which matches the kinematics of the corresponding real robot. Furthermore, pure virtual robots or other active objects such as carts can be visualized if the user defines appropriate kinematics. Figure 9 shows two visualized real robots and a pure virtual cart.

Table 2 lists commands of the Robotic Visualization Tbx. The first three commands are used to initialize, finish and monitor a visualization session. With the creation of a MATLAB 3D-figure at startup, a MATLAB timer object is also started. It ensures that the visualized objects are refreshed 20 times per second to achieve a smooth appearance of the animation.

ViSu.start	initialize the visualization, open an empty 3D window
ViSu.stop	stop the visualization, delete all virtual objects, close the figure
ViSu.info	display all virtual objects with their ID, type, position and additional information depending on object type
ViSu.create()	instantiate a robot object of type 'Kuka', 'Kawasaki' or user defined type
ViSu.repose_robot()	alter position of robot object identified by its id
ViSu.delete_robot()	delete a robot object identified by its id
ViSu.place_env()	instantiate an environmental object
ViSu.repose_env()	alter position of an environmental object identified by its id
ViSu.delete_env()	delete an environmental object identified by its id
ViSu.place_part()	instantiate a moveable object
ViSu.repose_part()	alter position of a part object identified by its id
ViSu.delete_part()	delete a part object identified by its id

Table 2: User Interface for Visualization.

The other commands offer identical functionalities for the three different types of visualizable objects: robot, environmental and part objects can be I) initialized, II) repositioned, and III) deleted during a simulation session.

Currently, the toolbox is being revised extensively to harmonize the user interface and improve software stability and robustness.

2.3 Integrated RCV Tbx usage example

Figure 5 introduced in Section 1 shows an example of multi-robot configuration. Notice that in that configuration we have integrated real robots from different vendors, i.e. from Kuka (Agilus, KR3) and Kawasaki (FS003N) and some virtual robots, too.

In this section we focus on an academic scenario where we have a robot control being applied only to some virtual objects. For this, it is necessary to start two instances of MATLAB; of these, one acts as the control PC (client) and the other is the visualization server. These MATLAB instances can either be located on different computers, as shown in Figure 5, or on the same computer as in the following example. The example includes two robots, the environmental object table and a test tube which is classified as a 'part'.

Listing 3 illustrates how MATLAB control and visualization instances can interact. MATLAB commands in line 1, 2, 4, 6, 7, 11 to 12, and 14 have to be executed on the visualization instance, while the indented lines 3, 5, 8 to 10, and 13 are control commands which have to be executed on the control instance.

```

1  >> ViSu.start;
2  >> ViSu.create('Kawasaki', 40000,...
                 [0,0,0,0,0,0]);
3      >> r1=robot('open', 'Kawasaki',...
                 'tcpip', 'localhost', 40000);
4  >> ViSu.create('Kuka', 40001,...
                 [500,500,0,0,0,0]);
5      >> r2=robot('open', 'Kuka',...
                 'tcpip', 'localhost', 40001);
6  >> ViSu.place_env('table.stl',...
                    [-800 0 0 0 0 0],'blue');
7  >> ViSu.place_part('test_tube.stl',...
                     [-800 0 400 45 0 0],'white');
8      >> rset(r1,'signal', [-9, 10]);
9      >> rmove(r1,'home2');
10     >> rmove(r2,'home2');
11 >> ViSu.info;
12 >> ViSu.delete_part(1);
13     >> robot(r1, r2, 'close');
14 >> ViSu.stop;

```

Listing 3: Code Example of Interaction of the two Parts of RCV Tbx for MATLAB/Simulink.

After a virtual robot has been created with the command `ViSu.create()`, the MATLAB prompt for the visualization instance is blocked until the TCP/IP connection to the control instance is established. This is accomplished when the appropriate robot (`'open', ...`) command is executed on the control instance which initiates a robot control object. The table, as well as the test tube, just exist virtually and are passive objects that are not controllable and therefore have no counterpart on control instance. After line 7 is executed, the visualization looks as depicted in Figure 10.

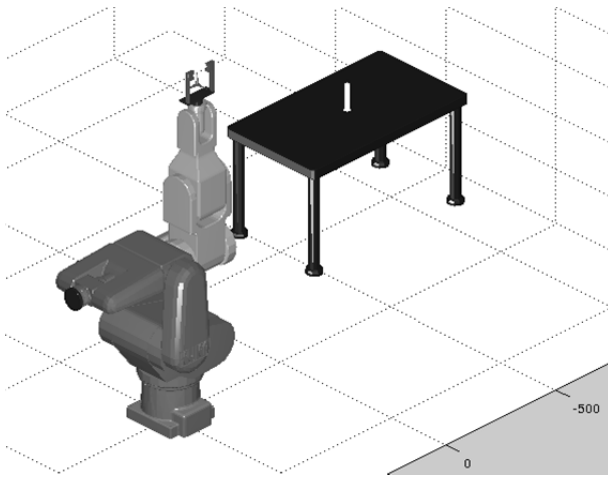


Figure 10: Visualization example after creation of objects.

Then some simple control commands, beginning with line 8, are executed. The command in line 8 closes the gripper of the Kawasaki robot. After that, both robots are moved to one of their predefined home positions, `'home2'`. Back on the visualization instance, information on the current visualized objects is requested. With this information the user knows, that the test tube is a `'part'` object with ID 1 and can be deleted during the visualization session. This feature is useful, if one wants, for example, to alter the surroundings of active robot objects without restarting the visualization.

After execution of line 12 the scenario looks as depicted in Figure 11. The gripper is closed, both robots have moved to their home positions and the test tube has disappeared. The last two commands close the TCP/IP connections and finally close and delete the visualization figure.

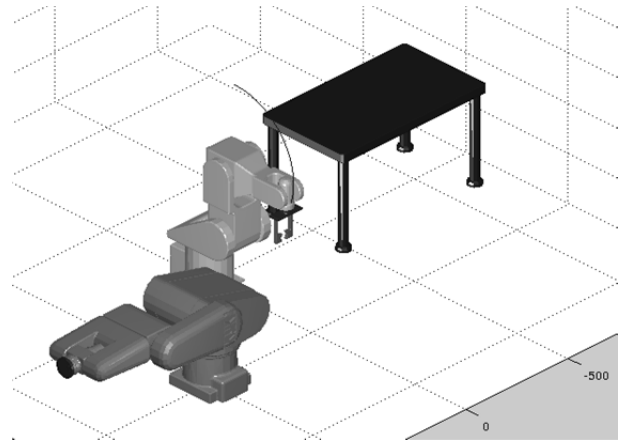


Figure 11: Visualization example after execution of some control commands.

Without larger modifications it is possible to apply the same sequence of control commands to real robots in a real environment, if we assume the real robots are of appropriate types and placed in the same positions. In this case, instead of the commands `ViSu.create()`, that initialize the visualized real robots, TCP/IP and RS232 connections to the control PC need to be established by the robot controllers. For visualized robots all connections are of type TCP/IP, although for a real KUKA KR3, for example, an RS232 connection is necessary. Hence, the control command in line 5 has to be adapted. The control object `r2` needs to be opened with the parameter `'serial'` instead of with `'tcpip'`. Instead of a visualized table and a visualized test tube, a real table and a real test tube could be placed in the real robot cell. Motivation for such simple control tests by simulating the movements first in a virtual environment could be, for example, to avoid collisions between robots and table.

3 Summary and Further Work

The RCV Tbx offers excellent possibilities for developing multivendor robot controls in a homogeneous software environment. It fulfills the main requirements of a RCP capable tool. The key benefits of using the RCV Tbx are the possibility to develop monolithic control programs for interactive robots from different manufacturers and the easy, manufacturer-independent integration of external hardware.

The MATLAB/Simulink environment is well established in the area of engineering and, today, is a standard tool for engineers. Hence, the engineer can benefit from employment of all programming tools available in this environment. Especially for recent advanced control applications such as the Simulation Based Control (SBC) approach introduced by Maletzki [9] the integration of the RCV Tbx into MATLAB/Simulink is very advantageous. Simulation models are directly used as control programs in this approach and it is obvious that RCV Tbx control commands can easily be integrated into these kinds of simulative control.

If one takes another step, it can be seen that models employed for control tasks can also be automatically generated from a knowledge base as proposed in the SBC and System Entity Structure (SES) approach by Schwatinski and Freymann [10, 11]. The development of flexible, task oriented multi-robot controls is considerably simplified with this approach.

References

- [1] Pawletta T, Freymann B, Deatcu C, Schmidt A. Robotic Control and Visualization Toolbox for MATLAB. In: Breitenecker F, Kugi A, Troch I, editors. *MATHMOD 2015*. Proceedings of MATHMOD 2015 - 8th Vienna Int. Conf. on Mathematical Modelling; 2015 Feb 18.-20., ARGESIM Report No. 44, ARGESIM, Vienna/Austria UT; 2015. P. 371-372 & Poster.
- [2] Research Group CEA. (2011). *Robotic Control & Visualization (RCV) Toolbox for MATLAB* [Internet]. [cited 2015 March 28]. Available from: http://www.mb.hs-wismar.de/cea/sw_projects.html
- [3] Abel D, Bollig A. *Rapid Control Prototyping – Methoden und Anwendungen*. Springer-Verlag Berlin. 2006.
- [4] Visual Components Corporation (2014). *3DRealize-R* [Internet]. [cited 2014 Oct 24]. Available from: <http://www.visualcomponents.com/products/3drealizer-r>
- [5] Chinello F, Scheggi S, Morbidi F, Prattichizzo D. KCT: a MATLAB toolbox for motion control of KUKA robot manipulators. In: *Proceedings of IEEE Int. Conf. on Robotics and Automation*, Anchorage, Alaska, 2010; P. 4603-4608.
- [6] Chinello F, Scheggi S, Morbidi F, Prattichizzo D. KUKA Control Toolbox. Motion control of robot manipulators with MATLAB. *Robotics Automation Magazine*, IEEE, 2011; 18(4):69-79.
- [7] Maletzki G, Pawletta T, Pawletta S, Lampe BP. A model-based robot programming approach in the MATLAB/Simulink environment. In: *Advances in Manufacturing Technology – XX*, 4th Int. Conf. on Manufacturing Research (ICMR06); 2006 Sept. 05-07; Liverpool, UK, 2006. P. 377-382.
- [8] Orth P, Bollig A, Abel D. Rapid Control Prototyping diskreter Steuerungen in der Automatisierungstechnik. In: *SPS/IPC/Drives Congress*; Nürnberg, Germany; 2004. P. 143-152.
- [9] Maletzki G. *Rapid Control Prototyping komplexer und flexibler Robotersteuerungen auf Basis des SBC-Ansatzes*. [dissertation in German]. Rostock University, Germany, 2014.
- [10] Schwatinski T, Pawletta T, Pawletta S. Flexible Task Oriented Robot Controls Using the System Entity Structure and Model Base Approach. In: *Simulation Notes Europe (SNE)*, 2012; 22(2): 107-114.
- [11] Freymann B, Pawletta T, Schwatinski T, Pawletta S. Modellbibliothek für die Interaktion von Robotern in der MATLAB/DEVs-Umgebung auf Basis des SBC-Frameworks. In: *Proceedings of ASIM-Treffen STS/GMMS*, Reutlingen 20./21.02.2014 - ARGESIM Report Nr. 42, ASIM Mitteilung AM 149, ARGESIM/ASIM Pub. Vienna, Austria, 2014, P. 199-208.

Acknowledgement

The authors are thankful to our former co-worker Tobias Schwatinski and all students who worked very motivated within the project. Furthermore, the authors acknowledge the financial support of the Federal Ministry of Education and Research and the Ministry of Education, Science and Culture of Mecklenburg-West Pomerania.

Towards a Newer Toolbox for Computer Aided Polynomial Design of Sampled-Data Systems

Rudy Cepeda Gomez, Bernhard P. Lampe*

Institute of Automation, University of Rostock, 18051 Rostock, Germany; *bernhard.lampe@uni-rostock.de

Simulation Notes Europe SNE 25(2), 2015, 79 - 84
DOI: 10.11128/sne.25.tn.10294
Received: August 15, 2015 (Selected ASIM STS 2015
Postconf. Publ.); Accepted: August 20, 2015;

Abstract. This work presents recent steps taken in order to update the DIRECTSD toolbox for MATLAB. This toolbox realizes recently developed polynomial methods for the analysis and optimal design of sampled-data systems. Once released, the version under development will be compatible with the newest versions of MATLAB and includes the possibility of working with both SISO and MIMO systems. The text describes the last published version, an interim version currently running and the updates planned for a future stable release. Some usage examples obtained with the interim version are also presented.

Introduction

Sampled-data systems are systems in which a digital computer controls a continuous-time plant. This class of systems is widely used in almost any industry. Due to the periodic sampling, sampled-data systems are periodically time-varying systems, so that the standard methods for time invariant systems cannot be applied. Classical approaches to the design of controllers for such systems consider either a continuous time synthesis of a controller followed by its discretization, or a discretization of the plant followed by a controller synthesis in the discrete time domain. Both approaches are approximations. Modern approaches to the design of optimal controllers are based on so-called direct design methods, which take into account the continuous time behavior of the system without approximations.

The widely known *lifting* technique [1] allows the transformation of some hybrid optimization problems to equivalent problems for time-invariant discrete systems. However, the dimension of the systems becomes infinite.

This methodology was implemented in MATLAB as the Sampled-Data Control toolbox [2, 3]. The lifting technique, however, is not applicable to some important cases, e.g., to systems with arbitrary time delays.

An alternative for the direct design of sampled-data systems is the frequency domain approach based on the parametric transfer function concept [4, 5], which allows to apply polynomial methods for analysis and design of sampled-data control systems. This approach has some advantages over the lifting technique: it can be used even when the continuous plant has time delays and it also allows to obtain the structure and order of optimal controllers.

The DIRECTSD toolbox, in its initial version [6], was designed to implement polynomial methods in the case of SISO systems. Its development generated a MIMO version [7], which uses the theory presented in [5]. Its latest version [8] was a mature project: albeit it focused on SISO systems, it included options to use either polynomial methods or the lifting technique to solve the problem.

In addition, a comprehensive set of examples and demos has been integrated into the MATLAB help explorer.

The development of the DIRECTSD toolbox, however, halted with its version 3.0. After almost ten years since its last revision, the code became obsolete, mainly because of the change in the object-oriented programming (OOP) model introduced in MATLAB release 2012.

During the last year, an effort has been made within the Institute of Automation at the University of Rostock to update the code and to have, once again, a functional toolbox. The steps taken in this direction and the current development status of the DIRECTSD version 4.0 are presented in this paper.

1 Structure of DIRECTSD 3.0

The DIRECTSD toolbox is currently in its version 3.0, which is available in [9]. The current code uses OOP to define a new class to create and manipulate polynomial objects. The class `poln` and its methods are one of the most important parts of the toolbox. To model continuous and discrete linear time invariant (LTI) systems, the standard objects from the Control Systems Toolbox of MATLAB, i.e. `tf`, `zpk`, `orss`, are used. Some functions for these objects were overloaded in order to change their behavior.

The toolbox provides functions to perform analysis and design of sampled-data systems using both the frequency domain methods presented in [4] and the lifting technique [1]. Due to some unsolved numerical robustness problems encountered during the development of the MIMOtoolbox only SISOcontrollers are considered in this version.

As the OOP model of MATLAB changed with version 7.6 (Release 2008a), the syntax of class `poln` made it obsolete. This change also removes the support of overloaded functions, making the methods newly defined for the standard objects unusable.

2 Interim: DIRECTSD 3.5

The first steps taken towards the newer version were aimed to recover a basic level of functionality while the new developer got familiarized with the structure of the code and its theoretical background. The result of this work is an interim version, which is being used as a basis for further developments.

For this intermediate version, designated as DIRECTSD 3.5, the class `poln` was rewritten to make it compatible with the new OOP model. To recover the functionality of the overloaded functions, new classes were defined as subclasses of the standard MATLAB classes. For example, for the `tf` class a subclass called `sdtf` was defined. In this way, the class `sdtf` inherits all the properties and methods of a standard `tf` object replacing the old methods with those that were overloaded in previous versions. Albeit this solution required extensive editing of the codes, it was the most practical way to replace the overloaded functions.

Besides the recovery of the basic functionality some minor extensions were added to the toolbox. The most important is the possibility of using a first order hold with systems affected by a time delay.

Since this version is considered as an interim while the newer version is under development, we have no plans to make it publicly available. However, any interested reader may contact the authors to request a copy.

The following subsections present some examples of usage obtained with DIRECTSD 3.5.

2.1 H_2 -Optimization of a system with delay using Zero and First Order Hold

This case study consists in solving the H_2 optimization problem for a sampled-data system affected by a time delay, using two different hold devices. The objective of the H_2 optimization is to find the transfer function $C(\zeta)$ of the LTI digital controller described in the complex variable $\zeta = z^{-1} = e^{-sT}$, such that the minimum value of the mean variance of the output $\varepsilon(t)$ is obtained. This mean variance is defined as

$$\bar{d}_\varepsilon = \frac{1}{T} \int_0^T d_\varepsilon(t) dt \quad (1)$$

where $d_\varepsilon(t) = d_\varepsilon(t + T)$ is the instantaneous variance of the signal $\varepsilon(t)$ [4].

We would like to compare the results (the order of the controller and the optimal cost) using each hold device.

The system under study is shown in Figure 1. It consists of a continuous plant $P(s)$, a digital controller $C(\zeta)$, a hold device $H(s)$, and a pure delay representing time needed for computation, communication or transport. The plant is taken as $P(s) = 1/s(s + 1)$, and a sampling period $T = 1$ sec is used. A value of $\tau = 0.1$ sec is set for the delay.

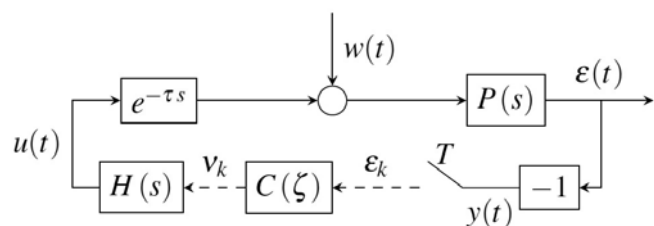


Figure 1. Structure of the system.

The two options considered for the hold device are a zero order hold (ZOH) for which

$$u(t) = v_k \text{ for } kT \leq t < (k + 1)T, \quad (2)$$

and a first order hold (FOH) for which

$$u(t) = v_k + (t - kT) \frac{v_k - v_{k-1}}{T} \quad (3)$$

for $kT \leq t < (k + 1)T$.

In order to use the DirecSD toolbox, the system must be converted into the standard form of MIMO sampled-data systems as shown in Figure 2.

Since in our case we have only a SISO system, all matrices and signals are scalar quantities.

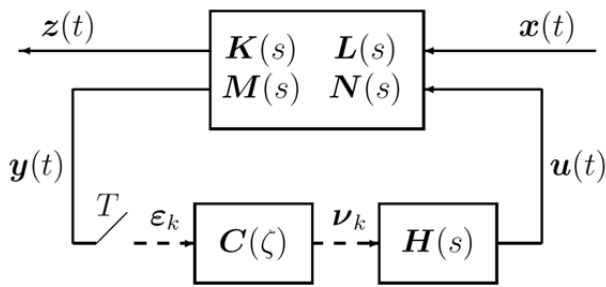


Figure 2. Standard sampled-data system.

Finally, we configure the scalar blocks

$$\begin{aligned} K(s) &= P(s), & L(s) &= P(s)e^{-\tau s}, \\ M(s) &= -P(s), & N(s) &= -P(s)e^{-\tau s}. \end{aligned}$$

The commands needed to run the example are as follows.

At first, let us set-up the system:

```
>>T = 1; %Sampling Period
>>H0 = tf(1, [1 0]); %Zero-order hold
>>H1(:, :, 1, 1)=ss(1); %First-order hold
>>H1(:, :, 2, 1)=ss(1); %First-order hold
>>P = zpk([], [0 -1], 1); %Plant
>>tau = 0.1;
>>Fdelay = tf(1);
>>Fdelay.iodelay = tau;
>>Pdelay = P*Fdelay;
>>sys = [P Pdelay; -P -Pdelay];
```

The last line creates a rational matrix, which represents a system in the standard form. The function `sdh2` is used to synthesize the controllers and to find the optimal cost.

The commands issued are

```
>>[Kzohd, err0d] = sdh2(sys, T, [], H0);
>>[Kfohd, err1d] = sdh2(sys, T, [], H1);
```

For the ZOH the toolbox reports the controller

$$C_0(\zeta) = \frac{-193.39(\zeta - 3.964)}{(\zeta + 171.1)(\zeta + 1.143)} \quad (4)$$

with a minimum cost of $\bar{d}_{\epsilon 0} = 0.6003$. On the other hand, using the FOH, we obtain the controller

$$C_1(\zeta) = \frac{200(\zeta - 3.819)}{(\zeta + 118.4)(\zeta - 2.705)(\zeta + 0.9415)} \quad (5)$$

with an optimal cost $\bar{d}_{\epsilon 1} = 0.6538$.

These results show that in this particular case the introduction of a FOH does not show any advantage over the use of a ZOH. Besides the higher order of the reported controller, the performance index worsens when the FOH is used. While a first order interpolator would give smoother results, here due to the necessary causality of the real hold element, the FOH becomes an extrapolator. This fact explains the above results. To have a more decisive comparison, Figure 3 presents the inter-sample variance $d_\epsilon(t)$ for both cases.

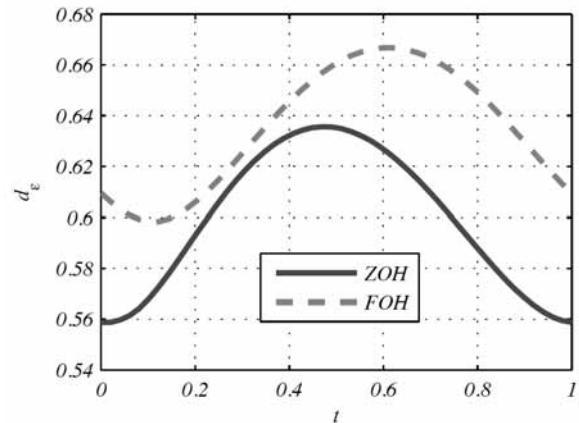


Figure 3. Comparison of the inter-sample variances for the cases with ZOH and FOH.

To obtain this plot, we used the function `sdh2norm` which finds either the mean variance or the variance at a certain point within the sampling interval.

For this particular case we issued the following commands to DIRECTSD 3.5

```
>>t = linspace(0, T, 50);
>>errs0d = sdh2norm(sys, Kzohd, t, H0);
>>errs1d = sdh2norm(sys, Kfohd, t, H1);
```

in order to obtain the variance at 50 equidistant points within the sampling interval. The results were then plotted to obtain the picture observed in Figure 3.

2.2 H_2 -Optimization of a generic system with two delays

This example considers the H_2 optimization of a closed loop sampled-data system, as shown in Figure 4.

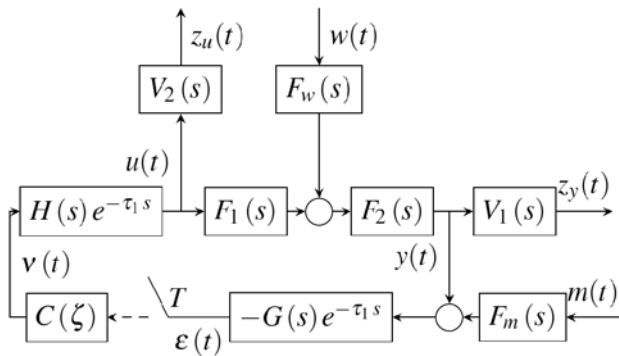


Figure 4. Structure of the system.

This is the most general case for a single loop system with a SISO controller.

For optimization purposes, the output of the system is taken as

$$\mathbf{z}(t) = \begin{bmatrix} z_y(t) \\ z_u(t) \end{bmatrix} \quad (6)$$

and the cost function to be minimized is

$$J = \bar{d}_z = \bar{d}_{zy} + \bar{d}_{zu} \quad (7)$$

In order to use the DIRECTSD toolbox, the system must be transformed into the standard form of Figure 2.

Realizing that the input of the system is

$$\mathbf{x}(t) = \begin{bmatrix} w(t) \\ m(t) \end{bmatrix}, \quad (8)$$

we obtain

$$\mathbf{K}(s) = \begin{bmatrix} V_1(s)F_2(s)F_w(s) & 0 \\ 0 & 0 \end{bmatrix} \quad (9)$$

$$\mathbf{L}(s) = \begin{bmatrix} V_1(s)F_2(s)F_1(s)H(s)e^{-\tau_1 s} \\ V_2(s)H(s)e^{-\tau_1 s} \end{bmatrix} \quad (10)$$

$$\mathbf{M}(s) = -[G(s)F_2(s)F_w(s)e^{-\tau_2 s} \quad G(s)F_m(s)e^{-\tau_2 s}] \quad (11)$$

$$\mathbf{N}(s) = -G(s)F_2(s)F_1(s)H(s)e^{-(\tau_1+\tau_2)s}. \quad (12)$$

Consider a simple case in which the disturbance and measurement noise are both white noise. This means that the forming filters $F_w(s)$ and $F_m(s)$ are both equal to 1. The weights of the output signals are taken as $V_1 = 1$ and $V_2 = 2$. The other components take the following values

$$F_1(s) = \frac{1}{s+1}, \quad G(s) = 1, \\ F_2(s) = \frac{2}{s+2}, \quad H(s) = \frac{0.25}{s+0.25}.$$

The two time delays are $\tau_1 = 0.05$ sec and $\tau_2 = 0.1$ sec, whereas the sampling period is taken as $T = 0.5$ sec.

This example is run by entering the following commands

```
>>F1 = tf(1, [1 1]);
>>F2 = tf(2, [1 2]);
>>Fm = 1;
>>Fw = 1;
>>tau1 = 0.05;
>>tau2 = 0.1;
>>G = -tf(1, 'iodel ay', tau2);
>>H = tf(0.25, [1 0.25], 'iodel ay', tau1);
>>V1 = 1;
>>V2 = 2;
>>T = 0.5;
>>H0 = tf(1, [1 0]);
>>K = [V1*F2*Fw 0; 0 0];
>>L = [V1*F2*F1*H; V2*H];
>>M = [G*F2*Fw G*Fm];
>>N = G*F2*F1*H;
>>sys = [K L; M N];
>>[C, err] = sdh2(sys, 1, [], H0)
```

Considering these values, the optimal controller is found as

$$C_{\text{opt}}(\zeta) = \frac{0.88946(\zeta - 1.284)(\zeta - 2.718)}{(\zeta + 3.721)(\zeta - 3.134)(\zeta - 13.81)} \quad (13)$$

with an optimal cost $J_{\text{opt}} = 0.99995$.

3 Future Version: DIRECTSD 4.0

A complete refurbishing of the toolbox is currently undergoing. We aim to recover the full capabilities of the last public release while we add some bells and whistles. Some of the improvements being added or planned to be added are

A new class for rational matrices: Up to this point, different classes have been used to represent polynomial matrices (class `pol n`) and rational matrices (linear systems in `zpk`, `tf` or `ss` form).

This created the need to some extra code to perform transformations when two objects of different classes should operate together.

The new classes will be subclasses of the standard MATLAB classes, so that they can inherit their properties and methods, and will also have all the particular methods to operate with polynomial matrices, i.e., rational matrices with denominator equal to one.

Packages for different methods: In its version 3.0 DIRECTSD includes options to work with the lifting technique or with polynomial methods.

These functions are being reorganized in the form of packages, according to the new OOP model of MATLAB.

Support for MIMO systems: The DIRECTSDM toolbox [7] was abandoned due to numerical instabilities of the MIMO version of the algorithms for polynomial design of sampled-data systems.

An effort is being made in to develop numerically reliable algorithms for this case. We are attempting to include said algorithms in the newer version.

Updated examples, demos and help files: The html help files will be revised and updated once the final code is in place.

Besides this user-facing improvements, the codes of all the functions are being revised. The coding style is being standardized and the internal documentation is being improved. These steps aim to simplify the maintenance of the toolbox.

For future versions, we are already considering the possibility of developing a graphic user interface, similar to the SISO tool included in the standard Control Systems Toolbox.

4 Concluding Remarks

This paper described the current development status of version 4.0 for the DIRECTSD, a MATLAB toolbox for the analysis and design of sampled-data systems. While an interim version, which recovered the basic functionality of the obsolete code, is in place, the work to obtain a completely revised version is on going.

Acknowledgement

The authors are thankful to Dr. K. Y. Polyakov for leaving the last version of the toolbox DIRECTSD 3.0, and for helpful discussions and explanations. The authors acknowledge the financial support of the German Science Foundation DFG.

References

- [1] Chen T, Francis B. *Optimal Sampled-Data Control Systems*. New York: Springer-Verlag, 1995.
- [2] Fujioka H, Yamamoto Y, Hara S. Sampled-data control toolbox: A software package via object oriented programming. *Proceedings of the IEEE International Symposium on ComputerAided Control Systems Design*; 1999 Aug; Hawaii, HI, USA; 1999, P. 404–409.
- [3] Fujioka H, Hara S, Yamamoto Y. Sampled-data control toolbox: object oriented of for sampled-data feedback control systems. *Proceedings of the IEEE Conference on ComputerAided Control Systems Design*; 2004 Sept; Taipei, Taiwan; 2004, P. 19–24.
- [4] Rosenwasser E Lampe B. *Computer Controlled Systems: Analysis and Design using Process orientated Models, ser.* Communications and Control Engineering. London: Springer-Verlag, 2000.
- [5] Rosenwasser E, Lampe B. *Multivariable Computer Controlled Systems: A Transfer Function Approach, ser.* Communications and Control Engineering. London: Springer, 2006.

- [6] Polyakov K, Rosenwasser E, Lampe B. DirectSD- a toolbox for direct design of sampled-data systems. *Proceedings of the IEEE International Symposium on Computer Aided Control Systems Design*; 1999 Aug; Hawaii, HI, USA; 1999, P. 357–362.
- [7] Polyakov K, Lampe B, Rosenwasser E. DirectSDM- a toolbox for polynomial design of multivariable sampled-data systems. *Proceedings of the IEEE Conference on Computer Aided Control Systems Design*; 2004 Sept; Taipei, Taiwan; 2004, P. 95–100.
- [8] Polyakov K, Rosenwasser E, Lampe B. DirectSD3.0 toolbox for Matlab: further progress in polynomial design of sampled-data systems. *Proceedings of the IEEE Conference on ComputerAided Control Systems Design*, 2006 Oct; Munich, Germany; 2006, P. 1946-1951.
- [9] Polyakov K. DirectSD: Package for the analysis and synthesis of digital control systems [Internet]. [cited 2015 May]. Available from: <http://kpolyakov.spb.ru/science/dsd.htm>

Simulating a Pneumatics Network using the Modelica Fluid Library

Patrick Drente¹, Peter Junglas^{2*}

¹Waskönig+Walter Kabel-Werk GmbH u. Co. KG, Ostmoorstr. 77, 26683 Saterland, Germany

²PHWT Vechta/Diepholz/Oldenburg, Schlesierstr. 13a, 49356 Diepholz, Germany; *peter@peter-junglas.de

Simulation Notes Europe SNE 25(2), 2015, 85 - 92
 DOI: 10.11128/sne.25.tn.10295
 Received: August 10, 2015 (Selected ASIM STS 2015
 Postconf. Publ.); Accepted: August 15, 2015;

Abstract. The Modelica Fluid library provides a framework and a large number of components for thermo-fluid applications. This should simplify the task of modeling a large pneumatics system considerably. Nevertheless a lot of problems remain. Some of them are due to the lack of important components that are difficult to model, others are related to deficiencies of the used modeling software. But the really hard problems come from conceptual difficulties that are well known to experts and plague the practical modeler. The lessons learned from building and using a basic pneumatics library show what is feasible right now - and where future work is needed to make modeling of fluid systems easier.

Introduction

Pneumatic systems are used in industrial applications to distribute power in factory buildings. Such installations can be huge, consisting of large distribution networks, several compressors and many different kinds of consumers. Planning a new network or optimising an existing one involves a large number of parameters and possible configurations. A particular problem is the unknown timing behaviour of the consumers, which is often highly irregular. This obviously calls for an appropriate simulation tool.

The simulation of fluid systems with compressible media is a difficult task. A large step forward has been the introduction of the Modelica Fluid library [1] (in the following often abbreviated as MFL).

It incorporates the fundamental behaviour of one-dimensional thermo-fluid systems and contains basic components for vessels, pumps, valves, pipes and other network elements. It uses the Modelica Media library [2], which allows to choose from a large list of predefined fluid media, compressible as well as incompressible.

Under these conditions Waskönig+Walter decided to start a simulation study in cooperation with the PHWT to optimise its existing pneumatics network. This simulation should help to find the reasons of bottlenecks and to evaluate the effects of simple actions beforehand. The tool chosen was OpenModelica, which is open source and has a good support of the MFL. From the academic point of view two questions were of special interest: How easy is it to use the MFL for development of own libraries? What is the current status of OpenModelica concerning MFL?

In a first step a simple library has been constructed that contains all necessary components. It does not compete with any of the commercial pneumatics libraries available, but concentrates on the basic network. Notably missing are models for valves and actuators. A special feature is a focus on the tee branch, which is a common device in pneumatics networks and is astonishingly hard to model with reasonable accuracy.

The choice to work with OpenModelica had serious consequences. To make clear which of the difficulties are intrinsic to the modeled system and which are due to deficiencies of the software, the construction of the library and the models will be described in the following using Dymola, which is arguably the best platform for MFL applications at present. The particular problems coming from using OpenModelica are subject of a later section.

1 The PneuBib Library

All components that are needed in the following to model simple pneumatics networks are combined in the PneuBib library. Two basic assumptions are made throughout: The temperature is constant and given by the ambient temperature, and the medium used is SimpleAir from the Modelica.Media library.

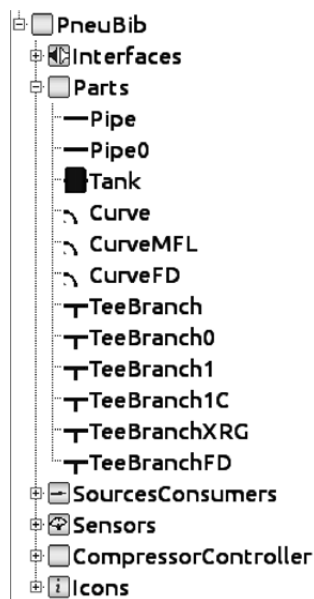


Figure 1: PneuBib library.

The model names are in German, but will be translated here for better readability. The library consists of the following packages (Figure 1):

- `Interfaces` and `Icons` provide the common infrastructure of ports, base classes and functions.
- `Parts` contains the main components for the network and will be discussed in more detail below.
- `SourcesConsumers` includes the usual source blocks to define pressure, mass or volume flow as well as a generic consumer block. This is simply a wrapper around the linear valve component from the MFL connected to the ambient pressure. For convenience a special consumer is added that opens periodically with additional parameters for number of periods and start time.
- `Sensors` provide access to the values of pressure, mass or volume flow in a network model.
- `CompressorController` contains components for the modelling of the controlled compressors that are used at Waskönig+Walter.

The `Partssublibrary` contains a few components that are wrappers around corresponding MFL blocks: The `Tank` is an isothermal version of the `ClosedVolume`, the `Pipe` consists mainly of a `DynamicPipe` with two nodes. The `CurveMFL` uses the `CurvedBend` from the MFL Fittings package. Unfortunately it doesn't work in `OpenModelica`, therefore a simpler version `Curve` has been added that relies only on the MFL function `dp_curvedOverall_DP` to compute the pressure loss.

Finally `Parts` provides several versions of a tee branch. They are the most complex part of PneuBib and will be explained in the following section.

2 Modeling a Tee Branch

The seemingly simple tee branch is difficult to model due to its several operational modes corresponding to the directions of the flows at its three ports: It can be used for splitting the main flow using the side branch either as one of the outgoing directions or for the incoming flow. One can join two flows with the combined outgoing flow either going straight or through the side branch. And for a compressible medium one could even use all three connections in the same direction, either outgoing or incoming.

The flow situations in all these cases are completely different - and always very complicated. Fortunately we are not interested in the exact flow but only in the overall pressure drops. Of course these depend on many details like the exact geometry of the pipes or the roughness of the inner pipe surfaces. But for our purpose of designing or analysing a pipe network, a simple approximation is often good enough. For this reason the PneuBib library contains several tee branch models with different levels of complexity and accuracy.

The MFL contains two models named `TeeJunctionIdeal` and `TeeJunctionVolume`, but they are not useful here, since they only describe the mixing properties, not the pressure loss in the junction.

Fortunately this gap has been filled by the free package `FluidDissipation` [3], which contains a lot of functions for heat transfer and pressure loss in many important cases, among them the tee branch. For the computation of pressure losses the library relies heavily on [4], a large compilation of knowledge in form of formulae, tables and graphs.

To implement these into numerically stable models they had to be complemented by a whole bunch of interpolation and regularisation formulae and combined with sophisticated schemes to discriminate between the different modes of the tee junction.

In addition to the necessary functions the Fluid-Dissipation library contains a ready-to-use example component of a tee branch. This has been incorporated into the PneuBib as TeeBranchFD, which adds only a wrapper to fix many of the parameters that are not used in the pipeline context. The TeeBranchXRG component is a simplified version that uses the pressure loss functions directly.

A simpler version is TeeBranch1, which is reduced in three ways: First it implements only the two modes that are used in the following, namely splitting and joining along the straight direction. Second, it neglects all density changes and uses only the density at the straight input for all computations. And finally it computes the pressure drops by using only simple interpolation polynomials for the pressure drop coefficients ζ_i as functions of the volume flow ratio $Q_{\text{branch}}/Q_{\text{combined}}$.

The task of finding an appropriate polynomial is not easy not because it is hard to find one, but because there are many published versions. Figure 2 gives an impression of the large variations in published values. The relations that are implemented in TeeBranch1 are based on data given in [5] for the split case and on the formulae in [6] for the join case.

Fortunately the general conclusions for the models of interest here do not depend significantly on the details of the chosen curve.

The largest impact of the simplifications has the assumption of constant density. Therefore PneuBib contains the variant model TeeBranch1C that uses the appropriate variable densities for computing energy balances. Only for the computation of the ζ values it sticks to the mentioned polynomials - mainly because better results for compressible media are not available.

When dimensioning pneumatic networks in practice one often uses a very basic approach to include the pressure losses of tee branches: At each outgoing junction one adds a ‘virtual’ substitutional pipe that reproduces the pressure loss of the tee branch [9].

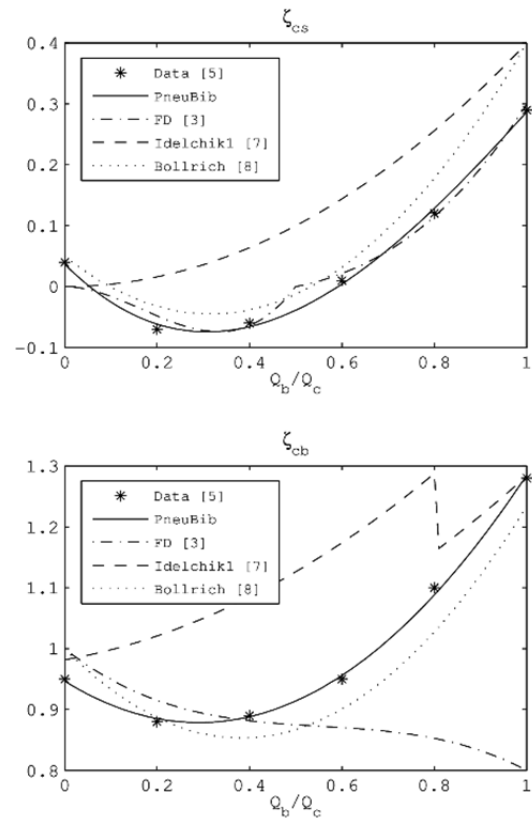


Figure 2: Pressure drop coefficient for the splitting tee branch.

By choosing an appropriate pipe length according to the dimensions of the junction one can get a rough approximation of the pressure losses.

Values for the substitutional length can be found from vendors of pneumatic equipment, e. g. at [10]. The basic component TeeBranch implements this idea.

3 Testing the Tee Branch Components

Several tests have been performed to check the basic functionality and to compare the different models of the tee branch. Figure 3 presents one of the test models. It is used to measure the pressure drops in the joining case, where a given time varying mass flow is inserted at the left port and a constant mass flow at the side port. The outgoing port on the right is connected to a fixed pressure.

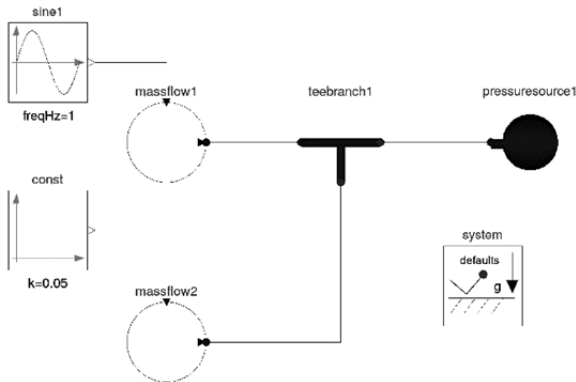


Figure 3: Model for testing a tee branch component.

The resulting pressure drops from the straight and the side connection to the output are compared in Figure 4 for the three models TeeBranchFD, TeeBranch1 and TeeBranch. The curves show a good qualitative agreement, their differences are to be expected considering the variability of the different underlying data.

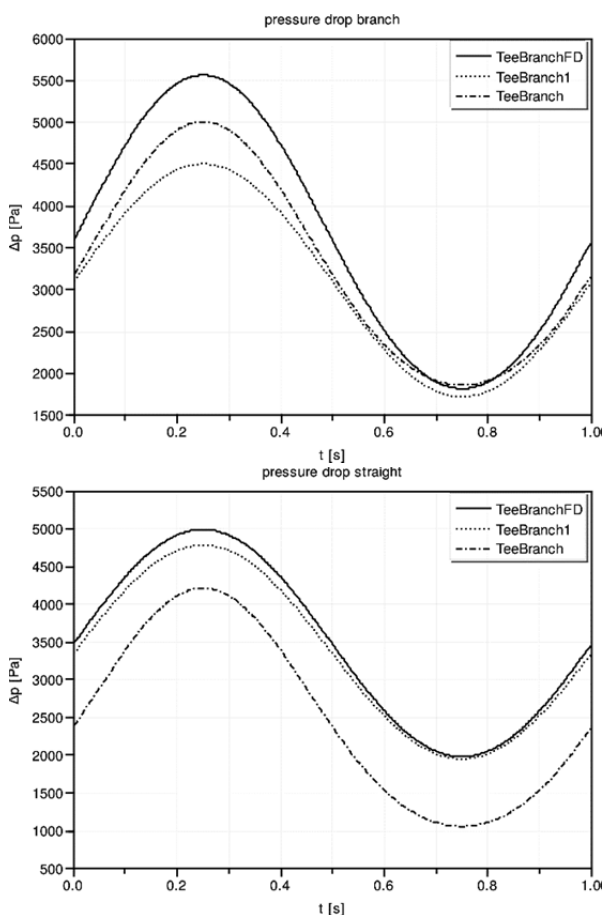


Figure 4: Comparison of the pressure drops in join mode.

One important thing that has been learned here is how to choose the parameters for the basic TeeBranch: The values given for the substitutional length vary by more than 25 % (e.g. between [9] and [10]), the roughness k of the substitutional pipe is not given at all. But the test results have shown that the roughness has a considerable influence. Therefore the lengths have been chosen according to [10], which includes values for the straight direction as well, and the value for k has been adapted to reproduce approximately the results of the other components. The resulting value of $k = 0.25$ mm seems reasonable being in the range for used steel pipes that is given in [11].

Similar tests in the split case have an unexpected behaviour: The elaborate models show an increasing pressure in the straight direction, whereas the simple TeeBranch gives a pressure drop.

A second thought explains this phenomenon: The pressure rise is the dynamical result of the velocity drop due to the splitting of the flow, an effect that is not incorporated in the simple ‘substitutional pipes’ model of TeeBranch.

But this doesn't make the simple model useless, there are two different ways how to cope with it: First one could just use it, including the pressure drop. After all the substitutional lengths (for the split situation!) are a reasonable rule of thumb coming from practical experience. Most likely it includes dissipative effects that have been neglected here so far. If one wants to reproduce the results of the other tee branch models instead, one can get a pressure rise just by using a negative length of the substitutional pipe.

It is rather unexpected that this simple idea works with Modelica's DetailedPipeFlow model, but so it does! In the following the TeeBranch will be used with parameter values that roughly reproduce the results of the other models.

To see how the different tee branch models perform as part of a network, in the next test one short straight pipe is inserted between the tee branch and the pressure source.

This leads to rapid pressure oscillations with amplitudes of several bar, which are strongly damped and converge to the expected result after a few seconds.

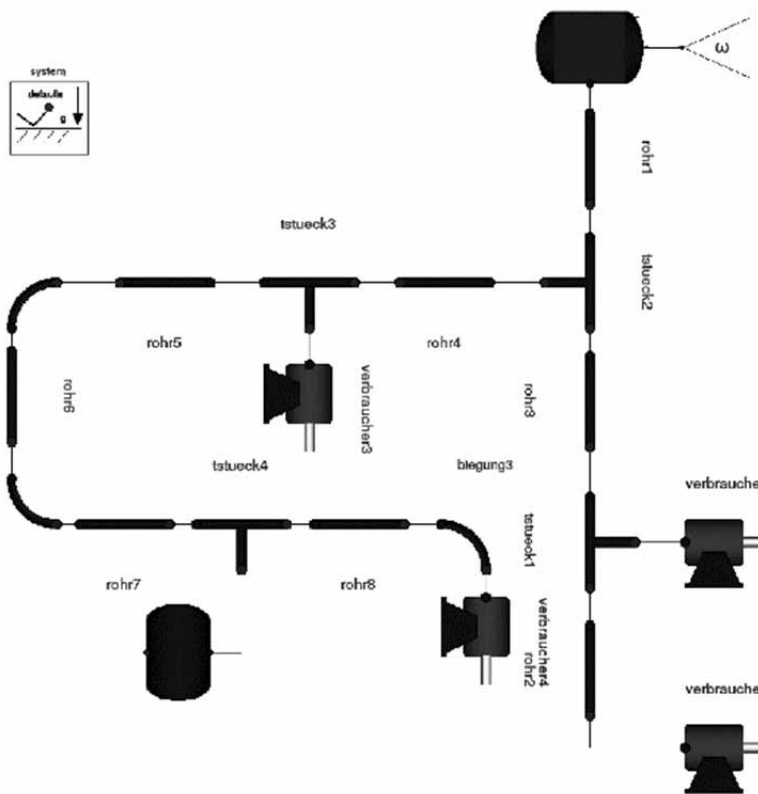


Figure 5: Model of a pneumatics network.

Their origin can be easily traced back to the start values: The `System` component defines a default start pressure for all blocks that is set initially to the ambient pressure of about one bar. Changing this to the value given by the pressure source reduces the amplitude of the oscillations to 0.2 bar. Setting the initial mass flow through the pipe to its steady-state value the amplitude goes down to 0.03 bar, which is in the order of the expected pressure losses. To get rid of the remaining oscillations too one had to supply more precise initial pressure values at all three ports, which are generally not known beforehand.

For the final test the position of the pipe has been changed: It is now inserted at the opposite side of the tee branch, directly connected to the mass flow source. In this case the simulation stops after a very short time with an error: The Newton solver is not able to produce reasonable initial values.

Trying to fix more initial conditions doesn't help at all, apparently very precise values are needed here for the solver to converge. The problem is the same for all tee branch models in `PneuBib`, even for the simple `TeeBranch`.

Only for the `TeeBranch` with positive substitutional pipe lengths the solver works fine and the results are as expected, but of course with a pressure drop along the tee branch instead of a rise.

A way how to cope with difficult initialisation problems has been proposed in [12]: One substitutes the problematic component with a simpler version that is used only for initialisation.

Using a homotopy, i.e. a continuous path from the simple to the complete model, one adapts the initial values gradually, until proper values for the final model are found.

This method has been applied successfully in different contexts, especially for thermo-fluid models [13].

According to this idea several simplified versions of the model have been developed, using a heavily reduced tee branch component, a pipe with a linear pressure drop law, a smoothly rising mass flow, a pressure source instead of one flow source or combinations thereof. All of them worked fine without the additional pipe, none of them lead to converging of the Newton solver. The only remaining tee branch model that actually works is the simple `TeeBranch` with positive substitutional pipe lengths.

4 Simulating Complete Networks

After the basic `PneuBib` library has been built and tested one can finally turn to the modeling of concrete pneumatics networks. A basic example consisting of several pipes, curves and branches together with a controlled compressor, a few consumers and an auxiliary tank is shown in Figure 5.

According to the findings of the last section all branches are simple `TeeBranch` components with positive substitutional lengths given by [10]. If one replaces just one of them with one of the more elaborated models the solver can't find initial values.

Nevertheless such a model can be used to answer some of the questions that appear in real networks, e.g.:

- How large are the pressure drops in several parts of the network and where are the bottlenecks?
- Can the compressor provide the necessary mass flows everywhere, especially in the case of two neighbouring consumers with large demands?
- What size and position should auxiliary tanks have to buffer peak demands?

As a concrete example the situation at the two consumers on the lower right of Figure 5 is considered. Their timing behaviour is shown in the upper graph of fig. 6: Both are used periodically with the same frequency, but different start time, leading to a small overlapping period, where both are working simultaneously. The diagram in the middle of fig. 6 displays the resulting pressures at both positions: When only one is used the pressure drops by about 0.15 bar, and it goes down by almost 0.4 bar, when they are working both. To resolve this problem an auxiliary tank can be installed between the two consumers. The resulting behaviour can be seen in the lower diagram of Figure 6. The total pressure drop is now reduced to almost half of the previous value.

Large systems with more than 60 components and 5000 equations have been studied in this way. In spite of the severe limitations of the tee branch model their results have been used successfully to improve a real pneumatics network.

5 Working with OpenModelica

As has been mentioned in the introduction all simulations should be performed with the open source program OpenModelica. Though its user interface is not as elaborated as those of commercial programs, its Modelica engine works generally very well in a wide range of applications [14].

Furthermore in the 1.9.1 release, which is the base of this investigation, it provides much better support of the Modelica Fluid library than most commercial programs except for Dymola. For this reason it seemed to be a cheap but promising alternative.

First tests using only MFL components showed promising results, most of the corresponding ‘wrapped’ PneuBib components like Tank and Pipe worked as well. Only the Curve showed a strange behaviour, the simulation lead to undefined values for the mass flow.

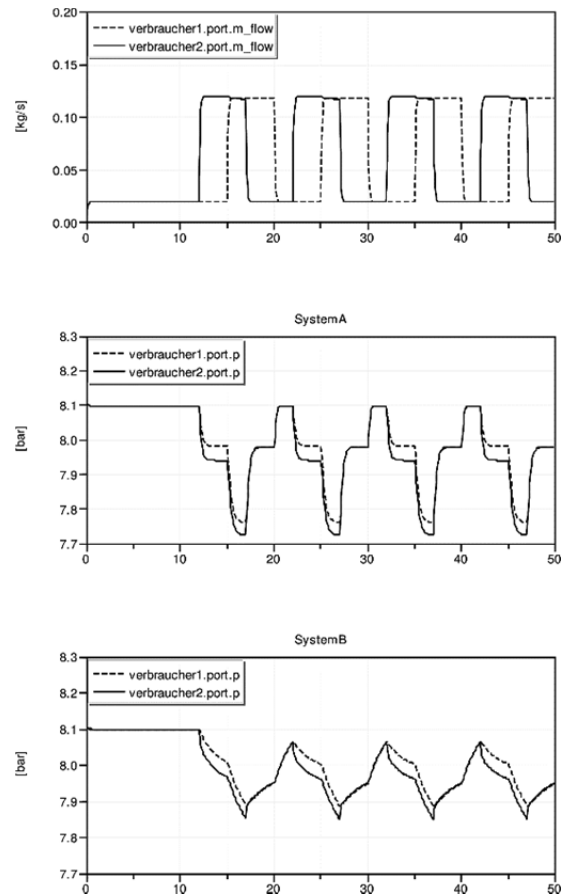


Figure 6: Simulation results of the example network.

Using instead the corresponding component BendFlowModel from the FluidDissipation library made things worse: Now the simulation lead to an error at initialisation. A working model could be constructed by simplifying the MFL version: Instead of using the function dp_curvedOverall_MFLOW to compute the mass flow from the pressure difference and taking into account the different densities and viscosities at both ends, it uses the inverse dp_curvedOverall_DP and the density and viscosity only at one end.

This worked in OpenModelica and lead to almost identical results in all cases of interest here (as verified later using Dymola).

The real challenge was to find a tee branch model that works in OpenModelica. Tests with the example component that is included in the FluidDissipation library did not succeed, the solver produced an error while flattening the equations. Simplifying this model by using the pressure loss functions directly did not remove the problem.

Only after reducing the complexity largely one arrived at a working component that is included in PneuBib as TeeBranch1 and described above.

That the structure of this model is on the brink of OpenModelica's capabilities becomes apparent, when one tries to remove some of its limitations: Using TeeBranch1C that includes some effects of the varying density, the simulation stops after a short time and produces very strange results that are due to completely wild initial conditions.

Since due to the initialisation difficulties all the problematic components could not be used in the final models anyhow, the prospects were good that the real pneumatics network could be analysed. Unfortunately in the case of the complete model with more than 100 components and 10000 equations OpenModelica gives up due to sheer size, the Modelica compiler crashes in an early phase.

Several simplified versions have been tried with nonconclusive results: Some models with 80 components and 7000 equations run for a short simulation time, before the compiler stops with an error, other much smaller models crash immediately.

A reduction of the network to 60 components and 5000 equations has led to a model that generally runs long enough to produce interesting results even after several structural modifications or parameter changes. It was the basis of the final investigations, which led to improvements of the real pneumatics system.

6 Conclusions

The modeling and simulation of the "simple" pneumatics network turned out to be much harder than had been expected at the beginning. This is a consequence of several different problems:

- The MFL components are very complex, they contain many parameters and subsystems that are difficult to understand for an unexperienced user. This problem gets much worse if one tries to create similar components from scratch.
- Some important components are missing, especially for the tee branch. The FluidDissipation library is a useful addition here, but it is not easy to use either. Especially the documentation and presentation of ready-to-use components leaves room for improvement.

- The fundamental problem of initialisation seems to be still far from being solved. Maybe the homotopy method is a feasible approach, but at least the authors were not able to find a working solution here.

The decision to use OpenModelica did not help either: Though it generally works fine in simple standard situations, it still has serious problems with models that are very complex or very large.

Especially it did not cope well with the FluidDissipation library. In combination with the intrinsic problems listed above this led to endless debugging sessions where it was almost impossible to isolate the reason of a specific failure.

Nevertheless in the end it all worked out: The industrial partner has now a working tool that helps to optimise his pneumatics installation. It has a sufficient accuracy and is open source. This miracle came about by simplifying the original system dramatically until all difficulties disappeared. Along the way one got rid of the limitations of the tool.

Is the MFL ready for end users? Principally yes - but only, if you are willing to accept simplified solutions and are prepared to work very hard!

References

- [1] Franke R, Casella F et al. Standardization of Thermo-Fluid Modeling in Modelica.Fluid. In: *Proceedings of the 7th Int. Modelica Conference*; 2009; Como; P. 122-131.
- [2] Casella F, Otter M et al. The Modelica Fluid and Media Library for Modeling of Incompressible and Compressible Thermo-Fluid Pipe Networks. In: *Proceedings of the 5th Int. Modelica Conference*; 2006; Vienna; P. 631-640.
- [3] Vahlenkamp T, Wischhusen S. Fluid Dissipation for Applications - A Library for Modelling of Heat Transfer and Pressure Loss in Energy Systems. In: *Proceedings of the 7th Int. Modelica Conference*; 2009; Como, P. 132-141.
- [4] Idelchik I E. *Handbook of hydraulic resistance*. 3rd ed. Mumbai: Jaico Publishing House; 2006.
- [5] Rodriguez ER. *Technische Hydraulik*. Lecture notes, FH Frankfurt [Internet]. [cited 2015 April 15]. Available from: <http://slideplayer.org/slide/209801/>
- [6] Aigner D. Gleichung zur Berechnung der hydraulischen Verluste der Rohrvereinigungen - kalibriert mit Ergebnissen numerischer und physikalischer Modelle. *3R-international*. 2008; 47(1).
- [7] Idelchik IE. *Handbook of hydraulic resistance*. Jerusalem: Israel Program for Scientific Translations; 1966.

- [8] Bollrich G. *Technische Hydromechanik, Band 1*: 7th ed. Grundlagen. Berlin: Beuth; 2013.
- [9] Bierbaum U, Hütter J. *druckluftkompendium*. 6th ed. Darmstadt: Weka Business Medien; 2004.
- [10] KAESER KOMPRESSOREN GmbH. *Berechnung des Druckabfalls* [Internet]. [cited 2015 April 15]. Available from: http://www.kaeser.at/Online_Services/Toolbox/Pressure_drop/default.asp
- [11] Bohl W, Elmendorf W. *Technische Strömungslehre*. 15th ed. Würzburg: Vogel Business Media; 2014.
- [12] Sielemann M, Casella F et al. Robust Initialization of Differential-Algebraic Equations Using Homotopy. In: *Proceedings of the 8th Int. Modelica Conference*; 2011; Dresden, P. 75-85.
- [13] Casella F, Sielemann M, Savoldelli L. Steady-state initialization of object-oriented thermo-fluid models by homotopy methods. In: *Proceedings of the 8th Int. Modelica Conference*; 2011; Dresden; P. 1-11.
- [14] Junglas P. *Praxis der Simulationstechnik*. Haan-Gruiten: Europa-Lehrmittel; 2014.

Efficient Modelling of Heterogeneous Battery Management Systems

Thomas Markwirth^{1*}, Matthias Gulbins¹, Karsten Einwich², Joachim Haase¹

¹ Fraunhofer Institut Integrierte Schaltungen, Design Automation Division, Zeunerstr. 38, 01069 Dresden, Germany; *thomas.markwirth@eas.iis.fraunhofer.de

² COSEDA Technologies GmbH, Königsbrücker Str. 124, 01099 Dresden, Germany

Simulation Notes Europe SNE 25(2), 2015, 93 - 98
DOI: 10.11128/sne.25.tn.10296
Received: August 10, 2015 (Selected ASIM STS 2015 Postconf. Publ.); Accepted: August 15, 2015;

Abstract. A simulation environment for the design of battery management systems (BMS) based on the modelling languages SystemC/SystemC AMS will be presented. Models of battery cells, the temperature and current sensors, the components for balancing of the battery cells, the integrated circuits to monitor the cell voltages and the interface for the data transmission to the controller where the BMS software is running describe the hardware. The software can be integrated via an AUTOSAR RTE compliant application programming interface into the simulation. The approach allows considering the nominal as well as the faulty behaviour of components. The simulation environment is integrated into the design environment COSIDE.

1 Introduction

The relatively low energy density of available electrical energy storage elements and their limitations for automotive applications currently form the critical point in the area of electro mobility. Therefore, the best use of battery cells regarding their capacity to extend the range of electric vehicles (EVs), increase of the life span of cells to decrease the costs of ownership, and the safe operation of battery cells are required. An optimal and intelligent battery management is essential to achieve these objectives.

The BMS must ensure the optimal charging and discharging of the battery over its lifetime and make sure that the battery never reached a critical state, which can lead to their destruction, fires or explosions.

Within the project IKEBA [1], a prototype of a virtual design platform is implemented, which allows to check the suitability and the interaction of selected hardware components (lithium-ion batteries and semiconductor circuits to monitor the battery status) and the software of a battery management system (BM software) by means of simulation. This way, the development of battery management systems including their software is supported. Investigations on how to increase the range of electric vehicles for different driving conditions by improvements of the battery management software and the characteristics of the applied hardware can be carried out using the design platform.

Partners in the project IKEBA are the Hella KGaA Hueck & Co, the Atmel Automotive GmbH, the Institute for Applied Materials - Applied Material Physics of the Karlsruhe Institute of Technology and the Fraunhofer-Gesellschaft e.V.

2 Implementation

The design environment COSIDE [2] for the development of electronic systems is the basis for the virtual simulation platform. COSIDE is a tool that allows handling of high complex electronic or heterogeneous systems together with the software that runs on these systems. Such systems can be modelled and simulated within COSIDE.

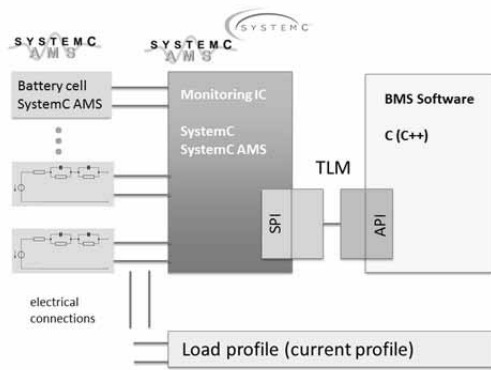


Figure 1: Components of the BMS Design environment.

The model creation for the hardware components is done with the SystemC and SystemC AMS hardware description languages. This approach allows a detailed inclusion of the components of the battery management integrated circuits (BM ICs) considering the characteristics of the Analog-to-digital converters (ADCs), delay times of the digital components, and the data interface (specifically SPI, the Serial Peripheral Interface) between BM ICs and battery management controller. The properties of batteries, circuits to compensate unequal state of charge of cells (cell balancing) and different load profiles can also be described. SystemC/SystemC AMS allows a compilation of the model descriptions for a fast simulation. For the development of the BM software an AUTOSAR RTE (Real Time Environment) provides a compliant application programming interface (API) with the required C function interfaces.

The virtual design platform is tested using a hardware demonstrator in the IKEBA project. The expected nominal behaviour is simulated and can later be compared with measurement results in order to validate the approach that is used for the virtual design platform. Later on, different concepts to operate the battery cells can be compared by means of simulation with the aim of increasing the range of an EV.

A number of models using SystemC and SystemC AMS have been created in connection with the modelling and simulation of the hardware demonstrator. A parameterization was done for the components that are to be used in the hardware demonstrator. Table 1 gives an overview of the currently existing models that have been implemented at Fraunhofer IIS/EAS. Through exchange and extension of models, the described solution can be easily adapted to other applications.

	Model	Description
Battery	battery_cell	Table model for battery cell
	cell_temp	Heat generation in the battery cell and exchange with the environment
	battery_pack_6s1p	Battery stack (6 battery cells in series)
BM IC	bmic_eln	Model of the electrical terminal behaviour of the BM IC for measuring and monitoring battery stacks in hybrid and electric vehicles
BM IC	bmic_simple	Behavioural model for BM IC for measuring and monitoring battery stacks in hybrid and electric vehicles.
Environment	simple_hvcs_eln	High voltage current sensor
	ntc_thermistor_characteristic	NTC resistance characteristic with temperature dependency
	temperature_measurement_eln	Resistance circuit for temperature measurement
	file_in_eln_isource	Source with reading a current profile from a file
BM Controller	RTE	AUTOSAR RTE compliant API of the BM controller (BM-software)
TLM	tlm2spi_target	Transaction Level Model (TLM) interface for SPI (for connection of the BM the BM IC controller)
	tlm_current_sensor	TLM interface for Local Interconnect Network (LIN) modelling (for connection of the power sensor to the BM controller)
	tlm_controller	TLM model of the BM controller

Table 1: Models for virtual design platform.

3 Example

The following Figure 2 shows the schematic and the corresponding non-electrical parts for a BMS system model, which was developed in the IKEBA project.

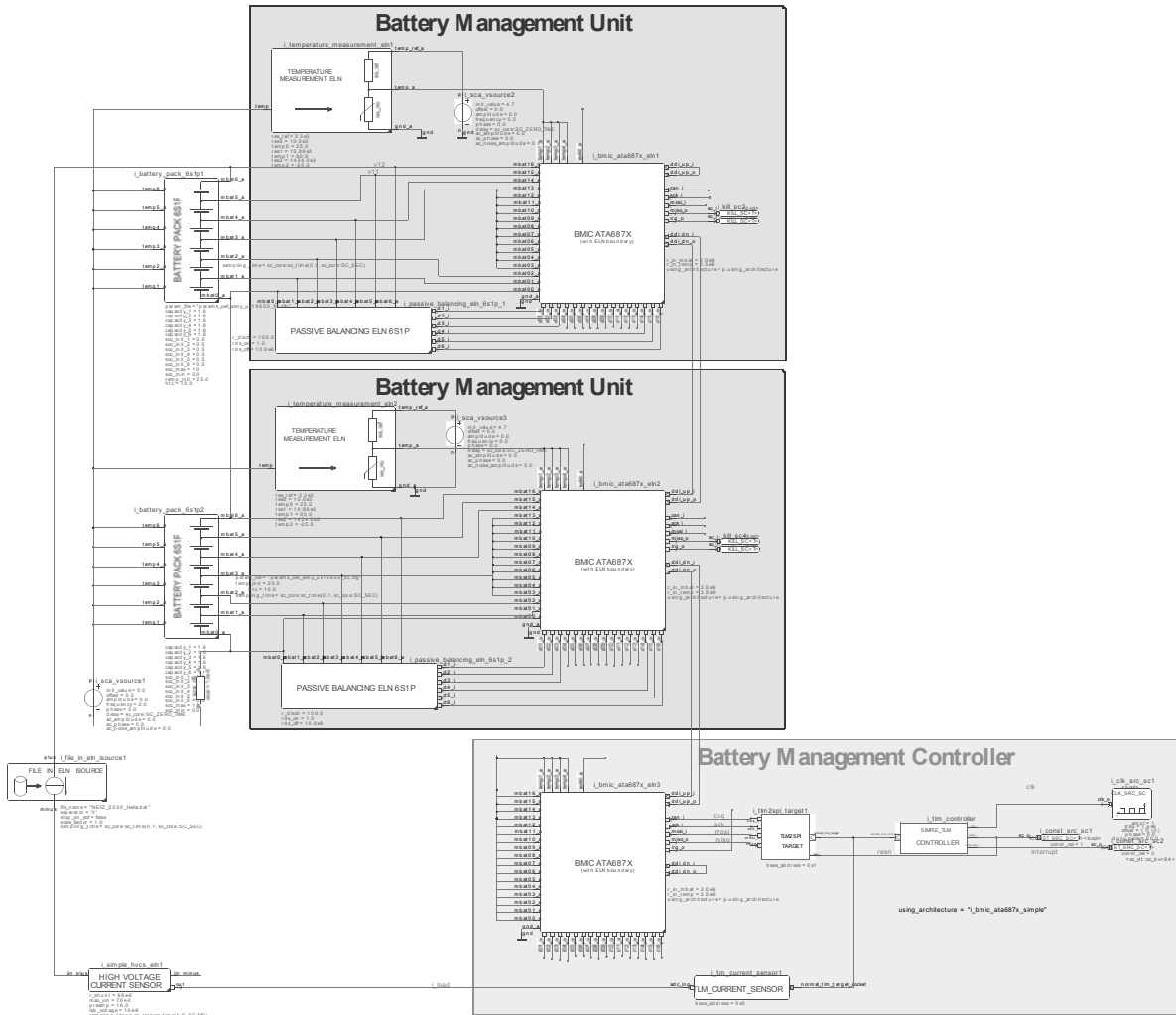


Figure 2: Top level description of a BM system model.

A generic approach, whereby individual components can be exchanged was applied here. A possible application is the investigation of the accuracy requirements of the ADCs for the voltage, current and temperature measurements. The function of the BM software regarding exchange of individual components such as battery cells and monitoring ICs can be checked. For investigations at system level a high performance of the simulation is mandatory to get meaningful results within a reasonable time. To ensure this, efficient modelling approaches as an abstraction of the data bus interfaces at the transaction level (TLM) are applied.

At this level, the focus is less on the physical implementation of the interface, the real signal realization, but more on the transaction of data itself. This supports a generic modelling approach, where changes of the implementation of the bus system can easily be realized.

This concerns the connection to additional components or the use of system components from other manufacturers. In conventional modelling approaches, modifications of the interface specifications require to change also the connected models. This would not be a generic approach. In automotive electronics, a variety of different interfaces for the connection of controllers with each other or to other circuits in the system is applied. In addition there are mostly misused variations of these interfaces. However, the exact connection of bus components and their modelling at low level plays usually only a minor role for functional system investigations. Furthermore, the TLM method improves the simulation performance and offers high efficiency, since the low level signal behaviour must not be reflected by the models. A high efficiency is also a prerequisite to perform statistical analyses or parameter variations.

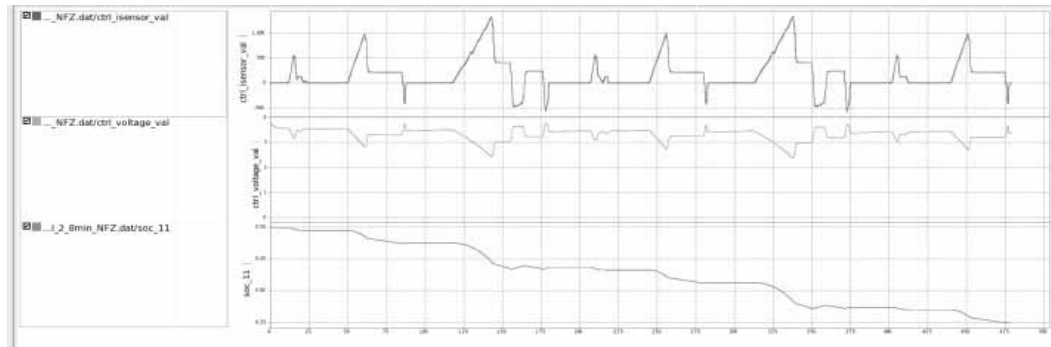


Figure 3: Current profile, voltage and SOC of a battery cell.

An example is given by the connection of the BM ICs with the battery controller and the inclusion of the BM software in the system description. Figure 3 shows simulation results for the battery voltage and SOC history based on a predefined current profile that corresponds to a NEFZ drive cycle for a special car.

4 Selected Models

4.1 Model for the battery cell

The batteries are composed of lithium-ion cells. The model of a single cell is described in SystemC AMS. The network model consists of an open circuit voltage V_{oc} , an internal resistance R_{IN} in series with two RC circuits $R_1 \parallel C_1$ and $R_2 \parallel C_2$, see Figure 4:

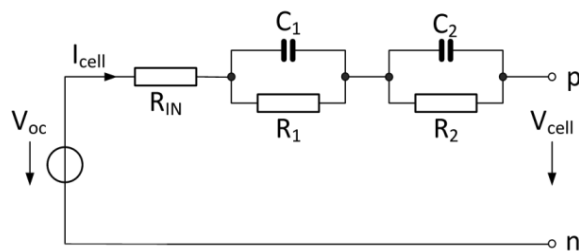


Figure 4: Structure of a cell model.

R_{IN} mainly represents the static internal resistance, the RC circuit $R_1 \parallel C_1$ models the regeneration after loading and unloading with a time constant of about a minute and the RC element $R_2 \parallel C_2$ contributes to the AC impedance with a time constant of about 1 ms. The elements of the equivalent circuit depend on the state of charge (SOC) of the battery cell and its temperature. The state of charge is updated w.r.t. the charge and discharge currents of the cell and the cell capacity that depends on the temperature.

The dependencies of the elements of the cell model can be determined based on measurement results for the loading and unloading of selected cells using the current interruption technique (CIT) [3]. For selected values of SOC and temperature, the corresponding parameters of the equivalent circuit model are determined. The dependencies are provided in the cell model by interpolation between table data points. Because the changes in SOC and temperature are "slowly", the model uses at the actual simulation time point their values at the last accepted time point. Thus, the evaluation of the battery model is a linear problem at each simulation time point. This circumstance helps to speed up the simulation. Analytical models to describe the dependencies require knowledge of cell parameters and make assumptions of functions that describe SOC and temperature dependencies. A widely used approach is suggested by Gao et al. [4] and also implemented in the design environment.

4.2 Interfaces of BM IC and BM controller

The interfaces of BM ICs and flow meter to the battery management controller have been described with transaction-level models (TLM). The following functions are supported by the transaction-level models:

- Initialization BM ICs
- Initialization of the high voltage current sensor
- Making available cell voltages using the BM ICs. If necessary separate functions for
 - Start the measurements
 - Read the values
- Making available cell temperatures using the BM ICs. If necessary separate functions for
 - Start the measurements
 - Read the values

- Making available currents through cells in series using the high voltage current sensor. If necessary separate functions for
 - Start the measurements
 - Read the values
- Control BM ICs signals for cell balancing

4.3 BMS application software interface

The BMS application software can access values provided by the BM ICs and the current sensor using compliant application programming interface (API) corresponding to an AUTOSAR RTE.

4.4 Inclusion of the BM - software

In order to include the battery management software in the system simulation, a controller model must exist which interacts with the hardware components. This is implemented by a generic model of the controller that calls the software applications with the help of a scheduler process. The scheduler process is sensitive to the external clock, reset and interrupt signals.

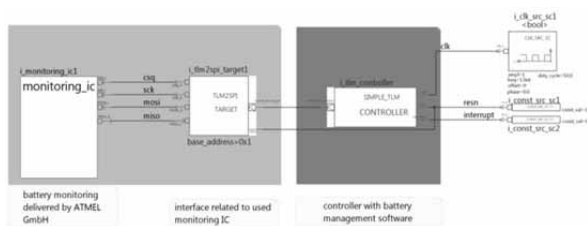


Figure 5: Interface with BM IC and BM controller.

Furthermore, access functions on the so-called hardware abstraction level are provided. These functions can be used by the software components. When calling such a function by a software application, a corresponding TLM transaction is initiated by the controller model. The transaction realizes the access to peripheral hardware. This approach provides an opportunity for high-performance access to hardware components. The access is independent of the actual physical interface. As another advantage, the software can be used for different architectures and peripheral configurations. It is only required that the selected hardware address is valid.

Thus, this solution enables a clear separation of the individual components. They can be developed consistently.

If the target hardware model has no TLM - interface, a conversion between the TLM representation and the real-world interface signals must be carried out. This was necessary for the BM IC model. For future applications, the converter model could be generated automatically based on the underlying fundamental basic libraries (e.g. TLM Sockets) and the specification of components (e.g. SPI frames for specific ICs).

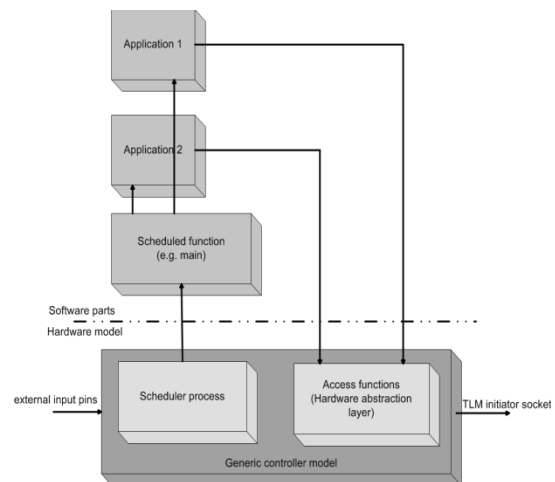


Figure 6: Generic controller model and implementation of software applications.

4.5 Hardware-in-the-loop application (HiL)

The system model can also be used for HiL (Hardware-In-the-loop) simulations. The SystemC/SystemC AMS [5] models are provided at a high abstraction level. This ensures a high simulation performance with sufficient accuracy, which is a prerequisite for modelling the real time use on a HiL system. While individual modules of the system are replaced by real, existing hardware components, other modules remain virtually. Potential benefit of this separation is the reduction of the simulation and model development time. Existing hardware components can be used together with models that run on a special HiL hardware. The models running on the HiL hardware emulate the corresponding components. Thus for instance, models can be used for components that are under development. This approach can also be applied to avoid problems with components that could be destroyed in the test process or that are dangerous or expensive.

A meaningful separation of real and virtual components of the system introduced in section 3 is to realize the battery, the BM IC behaviour, and current and temperature sensors using HiL hardware. The HiL hardware components are connected with the real BM controller. The BM software is running on the real BM controller.

The decision, to simulate the battery, is based on following basic benefits. A battery contains complex chemical and physical processes which are not reproducible and need time to adjust. SOC and temperature conditions can be modified in an easy way. There are high safety standards that must be fulfilled if a real battery is used for test purposes. Thus, real battery hardware should only be used in an advanced stage of development of the BMS. In the case of HiL simulations this is without meaning. Also the failure behaviour within the battery pack such as cable breaks, short circuits or intermittent contact can be realized easily within the HiL simulation. It is also possible to apply a number of drive cycles in the test procedure. Furthermore, the requirements for the components of the BM IC can be checked with a huge number of HiL simulation runs.

5 Outlook and Summary

The quality of a system also depends on the ability to react on incorrect or unexpected conditions. A system must be able to respond as expected and be robust to disturbances. Simulation allows investigating the system behaviour with faulty component behaviour. A library for the fault injection is prepared within the design framework. It contains generic structures to be able to reproduce incorrect behaviour in a simple manner. It is a special advantage of the approach that error descriptions are not part of the design under test (DUT). Error structures can be injected by the design environment to existing test benches without changing the test bench descriptions. This achieves a clear separation between system models and test environment.

Furthermore, the robustness of the system against parameter tolerances can be investigated using statistical analysis methods. The model descriptions based on SystemC AMS enable high simulation efficiency at system level. Libraries and experiences from other projects can be re-used for these tasks [6].

Simulation is an essential part of the development of complex heterogeneous battery management systems. It should support the investigation of the interaction of the battery, sensors, battery monitoring ICs, battery management controller and the application software. Compared to previous activities in this area, we developed an approach that allows considering all these parts in a simulation environment. It also supports the specification for specific hardware components as BM ICs. Consequences of hardware failures can be investigated in the test process in an easy way. The model development is based on SystemC/SystemC AMS. This approach opens the door to HiL hardware applications with models of the battery and electrical components straight forward.

References

- [1] Integrierte Komponenten und integrierter Entwurf energieeffizienter Batteriesysteme (Vorhaben IKEBA) [Internet]. Available from: <http://www.iam.kit.edu/awp/ikeba/>
- [2] COSIDE: The Design Environment for Heterogeneous System [Internet]. Available from: http://www.eas.iis.fraunhofer.de/content/dam/eas/de/documents/broschueren/fraunhofer-eas_coside.pdf
- [3] Lei B, Melcher A, et al.. Parametrisierung einer Ersatzschaltung für Li-Ionenzellen. *Poster Abstracts. Batterieforum Deutschland*, 2015 Jan; Berlin, Germany; 2015, P. MS11-VP19.2.
- [4] Gao L et al.. Dynamic Lithium-Ion Battery Model for System Simulation. *IEEE Transactions on Components and Packaging Technologies*. 2002;25(3): 495-505. doi: 10.1109/TCAPT.2002.803653
- [5] Einwich K, Arndt T. SystemC-AMS basierte Modellierung, Simulation und HiL Echtzeitsimulation für Anwendungen der Automobilelektronik. In: Liu-Henke, X. (Ed.): *Proc. ASIM-Workshop STS/GMMS*; 2012 Feb; Wolfenbüttel. Vienna: ARGESIM-Verlag. P.203–208.
- [6] Clauß C, Haase J, Markwirth T. Statistische Analyse mit dem SAE Standard J2748. In: Deatcu C, editor. *Proc. ASIM-Workshop STS/GMMS*; 2008 May; Wismar. Vienna: ARGESIM-Verlag.

Note. This work was supported by the German Ministry for education and research (BMBF) under grant 16N12440. The sole responsibility for the contents rests with the authors.

Design and Optimization of an Energy Manager for an Office Building

Kristin Majetta^{1*}, Christoph Clauß¹, Jürgen Haufe¹, Stephan Seidel¹, Torsten Blochwitz², Edgar Liebold³, Ullrich Hintzen⁴, Volker Klostermann⁵

¹Fraunhofer IIS EAS, Zeunerstraße 38, D-01069 Dresden, Germany; *kristin.majetta@eas.iis.fraunhofer.de

²ITI GmbH, Schweriner Straße 1, D-01067 Dresden, Germany

³NSC GmbH, Äußere Zwickauer Straße 8, D-08064 Zwickau, Germany

⁴FASA AG, Marianne-Brandt-Straße 4, D-09112 Chemnitz, Germany

⁵Provedo GmbH, Mottelerstraße 8, D-04155 Leipzig, Germany

Simulation Notes Europe SNE 25(2), 2015, 99 - 108
DOI: 10.11128/sne.25.tn.10297
Received: August 10, 2015 (Selected ASIM STS 2015
Postconf. Publ.); Accepted: August 15, 2015;

Abstract. Nowadays saving energy in the building sector is more important than ever. To achieve a reduction of energy consumption and also CO₂ emissions of buildings, energy efficient Building Energy Management Systems (BEMS), respectively called energy managers, are developed in the research project enerMAT. Therefore, enerMAT creates a novel approach for simulation, optimization, and verification that will aim to design a new generation of energy-aware optimized BEMS which will allow an overall cross-trade automatic control of energy flows to maintain user comfort whilst minimizing energy consumption and CO₂ emission.

The optimization referenced to energy uses a model-based approach with an overall building system model enabling the assessment of the energy performance for different design and operation alternatives of the energy manager in interaction with the building. This system model allows a simulation-based, energy-aware, global, dynamic, multi-criterial optimization of the energy manager. In this paper, the idea, the modeling of an office building and its energy manager and different optimization studies and their results are presented.

Keywords: Building, Energy Management, Green Building Library, FMI, PSO Optimization

Introduction

To save energy with the aim to not exhaust the not renewable sources (oil, gas, coal) is uncontradictedly one of the most essential tasks. One of the important energy “consumers” are buildings.

For heating, cooling, air conditioning, lightening and personal electronic devices, buildings consume more than one third of primary energy at all [1]. To decrease that part a lot of ideas were created. Besides passive measures (insulation) a promising way is consuming energy in buildings more ingenious than in the past. Energy shall only be consumed if it is absolutely necessary. One way to achieve this is the development as well as the introduction of Building Energy Management Systems (BEMS), shortly spoken, of energy managers. Operating on a runtime system energy managers take information from the building (via sensors) with the goal to calculate the energetically aware actions by controlling e.g. heating, cooling, air conditioning (HVAC).

At the ASIM workshop 2014 in Reutlingen-Rommelsbach we presented an approach of the development of energy managers which is investigated in the research project enerMAT [2]. Starting with an overall model of building, HVAC, ambience and user behavior an energy manager is designed (e.g. using state machines) which possesses some parameters. These parameters are used to adapt the energy manager in such a way that it is optimal regarding a cost function which bases on the energy consumption. The parameter adaptation employs optimization techniques. Once the energy manager is developed it can be transformed to a runtime system without any manual interaction.

In the enerMAT research project there are three demonstrator examples: a conference room, an office building, and a residential building. The above mentioned paper covers the conference room, whereas this paper is focused to the office building demonstrator.

The office building (Figure 1) is the headquarters of the enerMAT project partner FASA AG in Chemnitz. By applying the enerMAT approach the development of energy managers is shown.

1 Modeling

The research project enerMAT investigates a model-based approach which means that the development of energy management systems is done under utilization of a building model. Therefore, the modeling of the office building of the project partner FASA AG will be presented in this section.



Figure 1: FASA office building in Chemnitz.

The complex model of the FASA office building demonstrator comprises the building “itself”, underfloor heating, heat ‘generation’ (solar panel, boiler, heat pump, stove), control devices (UVR[3]), ambience (weather), user behavior, and the energy manager to be developed.

Modeling bases on the GreenBuilding library [4] which is developed by EA Systems Dresden. The library contains a large set of models for the development of energy systems with energy supply and storage systems, e.g. building component models, heating system models, heat pump models and ambience models. Since the GreenBuilding library is based on the free, object oriented, equation based Modelica language [5] it is possible to develop additional models that are required

in the project but are not part of the GreenBuilding library. The models are simulated using SimulationX that is developed by the ITI GmbH.

From energetic point of view the office building consists of two parts, the energy producing and the energy consuming part. The energy producing side (shown in Figure 2) consists of a solar heating system, a stove and a heat pump. All those components are heating up the water inside a 110 m³ buffer. The water of the buffer is temperature depending layered.

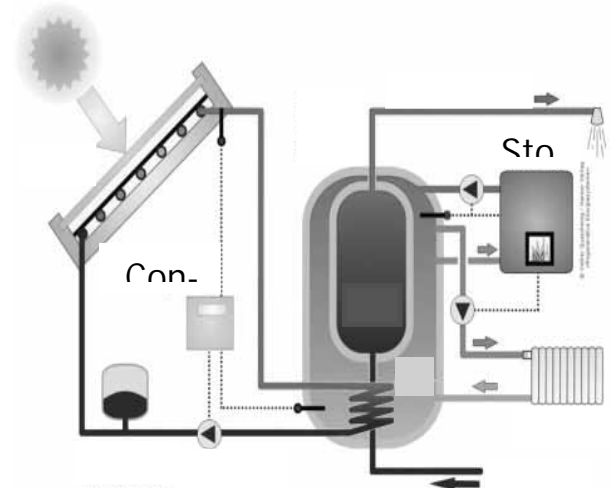


Figure 2: Energy producing part of the office building.

The energy generating system is controlled via the UVR1611 (Universalregelung 1611) which is a controller developed by the Austrian company Technische Alternative. To use the functionality of the UVR in the energy supply model, the UVR function blocks were modeled in Modelica and stored in a Modelica library.

On the energy consumption side the warm water of the buffer is used to heat the offices in the building via a floor heating system. The offices are modeled with the BuildingZone model of the GreenBuilding Library. The BuildingZone is a prepared, complex one-point model of a room, which calculates the heat losses taking ambience conditions, heat sources and losses into account. The modeler has to parametrize the zone with e.g. wall properties, room size or geographical direction. Similar office rooms were combined to a bigger building zone. The so developed model of the consumption side contains 25 building zones and needs about 5 hours for a one-year-simulation. Since this is far too long for later optimization, the 25 building zones were combined to two building zones which are representing the ground floor and the first floor of the office building.

The models for the energy generating and the energy consuming part are connected to the complete building model which helps developing the energy manager for the office building.

To round the complete building model off, a model representing the users' behavior is added. This model gives information about the presence of the employees on working days. This knowledge is used to specify users thermal comfort ranges that's compliance is ensured by the energy manager.

2 Energy Manager Development

By means of the complete model of the office building two building energy management systems are developed, one for the energy producing side and one for the energy consuming side. This separation is reasonable since the aim of the energy supply side is to maximize the solar earning over the year and the aim of energy consuming side is to minimize the energy usage.

A BEMS (energy manager) prescribes set points as well as parameters of local controllers by keeping their structures. In the literature several kinds of BEMS are known e.g. rule based BEMS[6], BEMS that are based on artificial neural networks [7], fuzzy logic [8], optimization [9] or ontologies [10] and context-aware BEMS[11].

enerMAT prefers UML statecharts (Unified Modeling Language) as a basic design approach of BEMS since states are induced naturally, like, in case of a temperature, 'too cold' or 'too hot'. Since the aim of the BEMS is controlling the set points of local HVAC devices, at each state of the statechart valid set points have to be calculated.

2.1 Energy manager for energy consumption

Most of the energy is consumed in the office rooms. Since minimizing the whole energy consumption will be achieved by minimizing the energy consumption of a single room, the energy manager is developed for one office room. Two different approaches were investigated.

The first approach is to work with lookup tables that contain all states that the room can have under defined surrounding conditions and values of the controlling variables to achieve a certain goal.

Such a table could, for example, contain temperature ranges for the ambient temperature and for the start value of the room temperature. The aim would be to know when to turn on the heating system to achieve a certain room temperature which means, it must be known how much time the room will need to reach the target temperature.

An example is shown in Table 1. Although the values in Table 1 fictitious, they are inspired by real values which are results of estimating another one-zone-room model developed with the GreenBuilding Library.

Ambient temp.	Start room temp.	Heating up time
< -5 °C	< 0 °C	10 h
	0 °C – 10 °C	8 h
	10 °C – 20 °C	5 h
	> 20 °C	4 h
- 5 °C – 15 °C	< 0 °C	8 h
	0 °C – 10 °C	7 h
	10 °C – 20 °C	4 h
	> 20 °C	3 h
15 °C – 30 °C	< 0 °C	7 h
	0 °C – 10 °C	2 h
	10 °C – 20 °C	1 h
	> 20 °C	30 min
> 30 °C	< 0 °C	3 h
	0 °C – 10 °C	2 h
	10 °C – 20 °C	30 min
	> 20 °C	10 min

Table 1: Example lookup table.

During the energy managers' runtime it permanently compares the actual room conditions with the conditions in the lookup table and decides permanently whether to switch on the heating system or not.

The obvious drawback of this method is the huge amount of data that is needed already for relatively small use cases such as the investigated single office room. Besides the ambient temperature and the start value of the room temperature, there is a number of other influence values like the temperature of the adjacent rooms and the supply temperature of the floor heating system.

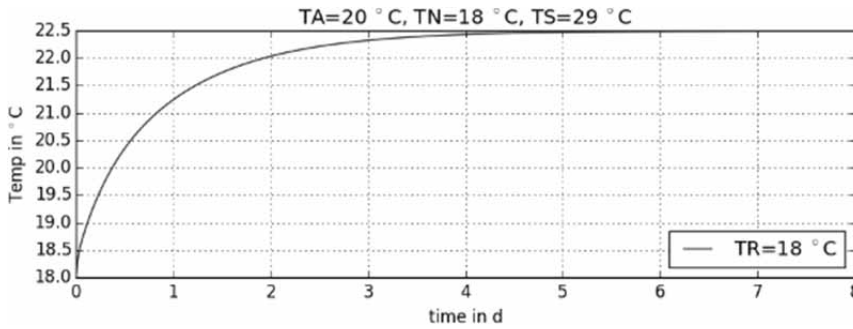


Figure 3: Heating up curve of single office room.

Before developing the lookup table its data items must be known. Therefore a great number of simulations had to be done to investigate the heating up time for each combination of ranges of the influence variables. Since this way of developing an energy manager for the office room seems to be not really feasible, a second investigation was done and is shown in the following.

The simulation of one heating up process of a single office room shows, that the heating curve has the shape of an exponential function $f(t)$ (Figure 3). $f(t)$ is characterized by its steady-state value g , its start value s and its slope $a(t = 0)$ and can be described by (1):

$$f(t) = g - (g - s)e^{-\frac{a}{g-s}t} \quad (1)$$

The variables g and a depend on the supply temperature $T_S(t)$ (TS in Figure 3), the start value of the room temperature $T_R(t)$ (TR in Figure 3), the ambient temperature $T_A(t)$ (TA in Figure 3) and the temperature of the adjacent rooms $T_N(t)$ (TN in Figure 3).

For simplicity reasons the temperatures of the neighbor rooms are chosen to be identical.

The variables g and a are identified using results from particular simulations with defined values of $T_S(t)$, $T_R(t)$, $T_A(t)$, and $T_N(t)$.

Using as an example g , the identification process is shown exemplarily. To identify the dependency of the steady-state value g from $T_A(t)$, $T_N(t)$ and $T_S(t)$, the following linear approach was chosen:

$$g = T_A \cdot x_1 + T_N \cdot x_2 + T_S \cdot x_3 \quad (2)$$

Since (2) has three unknowns (x_1, x_2, x_3), three simulations under defined conditions were undertaken and the linear system $Ax = b$ with

$$A = \begin{bmatrix} T_{A,1} & T_{N,1} & T_{S,1} \\ T_{A,2} & T_{N,2} & T_{S,2} \\ T_{A,3} & T_{N,3} & T_{S,3} \end{bmatrix}$$

$$\text{and } b = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix}$$

was solved. Using (2) for each value of T_A, T_N , and T_S , g can be calculated. A similar approach is applied for identifying the parameter a .

The parameter s does not have to be identified since it is the start value of the room temperature which simply can be taken from the simulation. Knowing g, s and $a(t = 0)$, (1) is parameterized during the whole simulation of the room model and calculates the points in time for turning on or off the heating system online by transforming (1) to

$$t_{on,off}(t) = -\frac{g - s}{a} \ln\left(-\frac{w - g}{g - s}\right) \quad (3)$$

where w is the desired target temperature in the room.

The cooling down process of the room after turning off the heating system is also described as an exponential function in the same way as the heating process is described. For technical reasons, the number of turning on and off the heating system is restricted to one per day.

Figure 4 shows the heating and cooling process of the room for a five day periode. In the first two and a half days the heating system is always turned on since the target temperature w (blue dashed curve) is not reached.

In the second two and a half days the heating system is turned on and off in a way that the room temperature (red, continuous curve) reaches the target temperature exactly at the beginning of occupancy and falls below the target temperature at the end of occupancy.

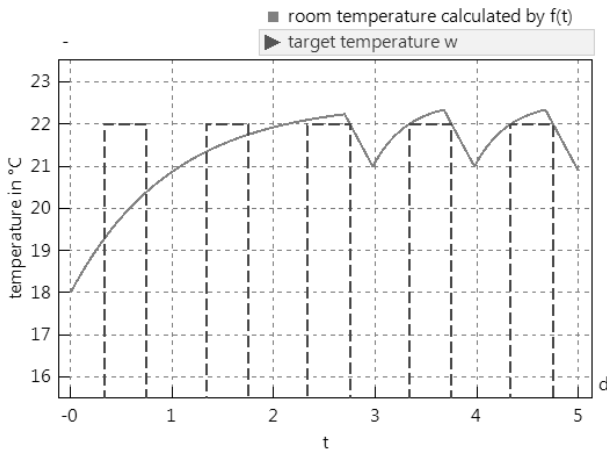


Figure 4: Heating and cooling of the office room (room temperature calculated by $f(t)$).

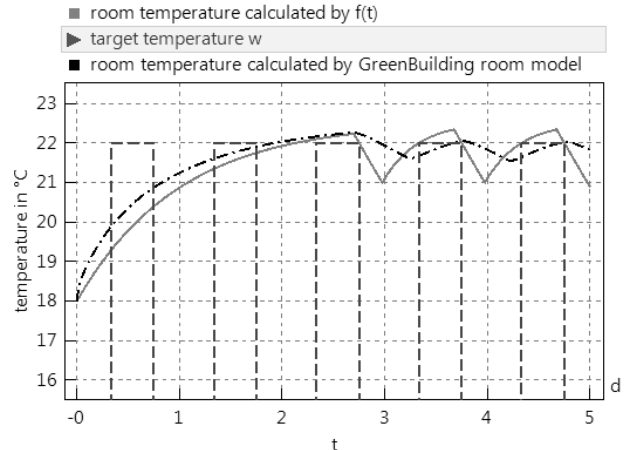


Figure 6: Comparison of room temperatures calculated by $f(t)$ and by the GreenBuilding room model.

The switching points for the heating system belong to the room temperature that is calculated by the exponential function (1). Since the aim is to develop an energy manager for a ‘real’ office room of the office building, in the following the calculated switching points are used to control the heating system of the GreenBuilding room model.

The results in Figure 6 show, that the room temperature of the GreenBuilding model (black dashed dotted curve) shows the expected behavior by trend. This means, that the heating period in the first two and a half days is nearly the same as for the room temperature calculated by $f(t)$ (continuous red curve) and that also the switching of the heating system can be seen in the second two and a half days.

Anyway the room temperatures are not exactly the same. A reason for that is that the exponential function (1) does not exactly represent the heating and cooling process of the room. That is because the slope of the ‘real’ process cannot be represented by simply parametrize the exponential function (1) with a constant slope $a(t = 0)$. In fact a slope that varies in time needs to be used.

Figure 5 shows the process of the energy manager development in a schematic way.

To achieve further energy savings the lowering of the target temperature in the office room on weekends was investigated. Therefore the target temperature on weekends was set to 18 °C (see Figure 7).

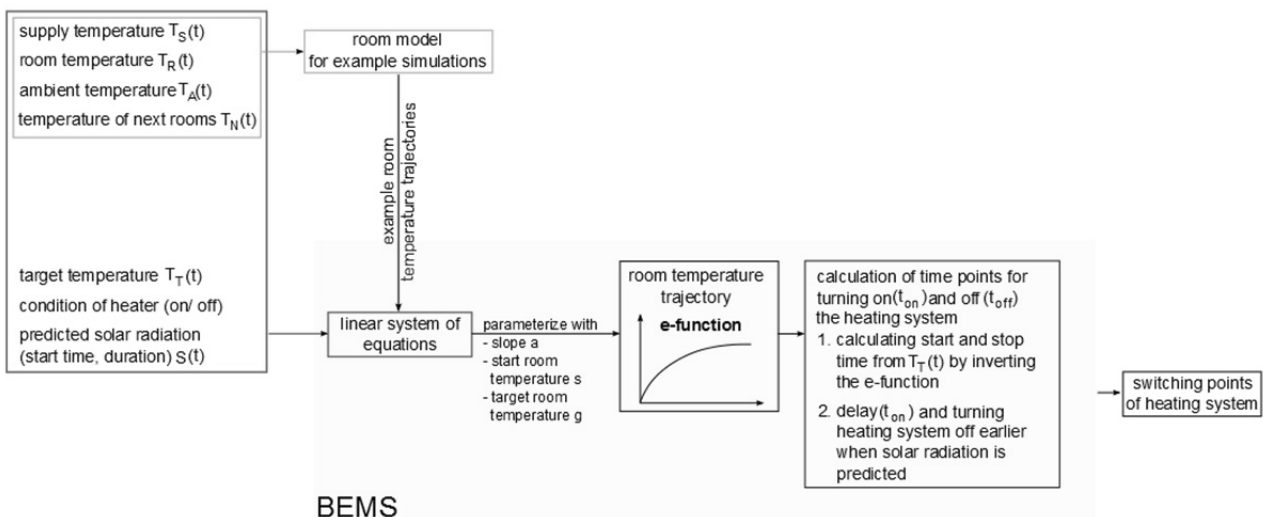


Figure 5: Schematic description of energy manager development.

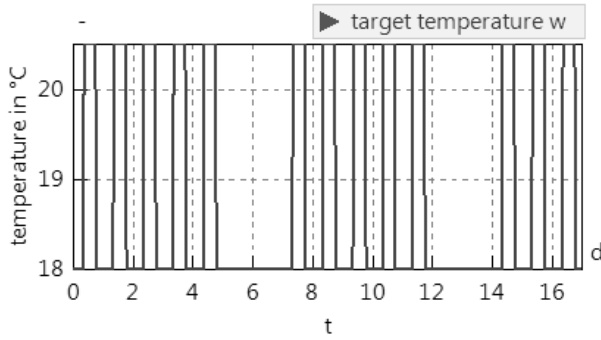


Figure 7: Target temperature w with lowering at weekends.

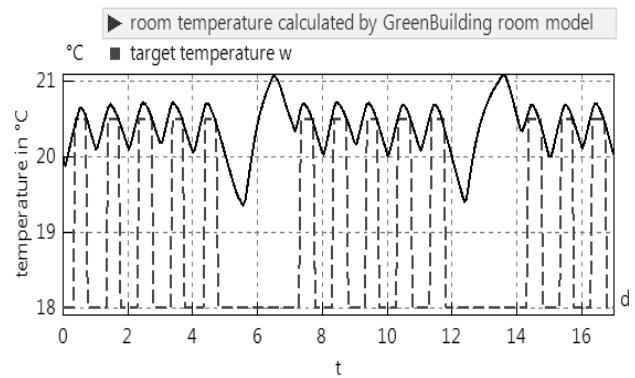


Figure 9: Temperature trajectory with setpoint lowering on weekends.

The switching points of the floor heating system calculated by $f(t)$ lead to the temperature trajectory shown in Figure 9. It can be seen that the room temperature has a very good gradient during the week but after a weekend the heating starts too early and therefore it gets too warm during the second half of the weekend. For that reason an optimization of the point in time for turning on the floor heating is described in Section 3.1.

The investigations on the energy manager were all done without taking solar heating into account. The next steps must be to analyze the time delay that the heating system can be switched on later and/or off earlier when the sun is shining into the office.

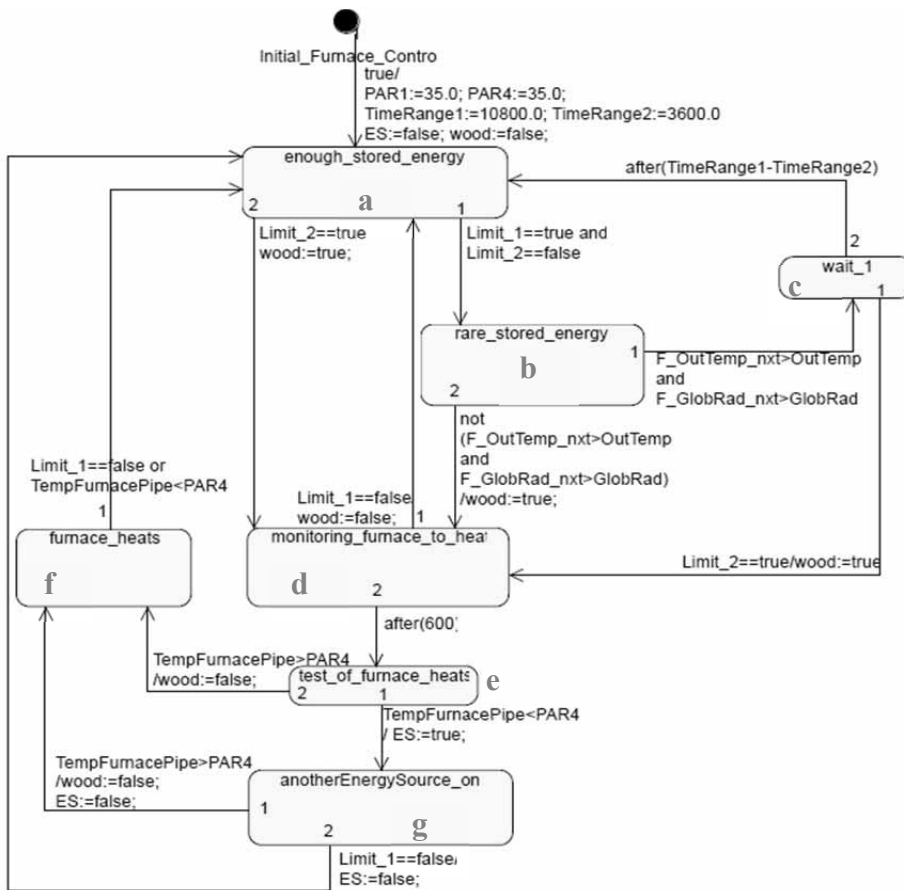


Figure 8: Statechart for controlling the stove.

Also the prediction of the ambience conditions such as temperature and solar radiation need to be taken into account in the future. Depending on the weather conditions, the actual and predicted occupancy and the actual room temperature, the points in time for turning on and off the heating system will be optimized.

2.2 Energy manager for energy generating

The solar heating is already controlled by the UVR1611 controller and does not need an energy manager. Missing is the information about starting the wood-burning stove when the water in the buffer is getting too cold.

The energy manager covering that task consists of a statechart named `furnace_ctrl` (Figure 9).

The decision whether the stove is turned on depends on the actual and predicted ambience conditions like temperature and global radiation and the needed supply temperature of the floor heating system.

If enough energy in the buffer (state a) is available, limits are checked to find out if the available energy becomes low or if heating the stove is necessary immediately. If energy is low (state b) and solar radiation is expected, it is sensible to wait (state c). If otherwise heating is needed (value *wood* is true, state d) it is checked whether heating has actually started (state e). If so (state f) it again is checked if enough heat is in the buffer (state a). If the stove has not started heating, it is checked if there is another energy source (state g) that could be used to heat the water in the buffer.

To calculate the Boolean limit values (limit 1, limit 2) which indicate if there is enough heat in the buffer, a reference temperature T_{ref} is computed (4) with the help of the adjusting parameters c_0 , c_1 (that will be optimized later) and the outdoor temperature T_{out} .

$$T_{ref} = \min(80, c_0 + c_1(25 - T_{out})) \quad (4)$$

The Boolean limit values are calculated by using T_{ref} with several underfloor heating supply temperatures T_{s1} , T_{s2} , T_{s3} and T_{s4} measured at the tank.

$$\begin{aligned} \text{limit1} = & (T_{s1} < T_{ref} + 5 \text{ and } T_{s2} < T_{ref}) \\ & \text{or} \\ & (T_{s3} < T_{ref} + 5 \text{ and } T_{s4} < T_{ref}) \end{aligned} \quad (5)$$

Further adjusting parameters c_2 , c_3 (that also will be optimized later) are used to calculate the burn time Bt , which is proportional to the mass of wood to be burned:

$$Bt = c_2 + c_3(25 - T_{out}) \quad (6)$$

This developed energy manager for the heat generating side of the office building is simulated and optimized together with the developed building models as shown in section 3

3 Optimization Studies

In this section different optimization studies and their results are shown. For the optimization the energy managers for generation and consumption are combined with a building model to complete simulation and optimization models.

These models are exported as functional mockup units (FMUs) and connected to a particle swarm optimizer via the functional mockup interface (FMI). Different optimization studies and their results are shown in this section.

3.1 Energy manager for energy consumption

As mentioned above the exponential function (1) does not exactly represent the heating and cooling process of the model room. This is why the points in time for turning on and off the floor heating system should be improved by optimizing.

Therefore two optimization parameters opt_{p1} and opt_{p2} were introduced that increase or decrease the time between turning the heating system on or off and reaching the target temperature w .

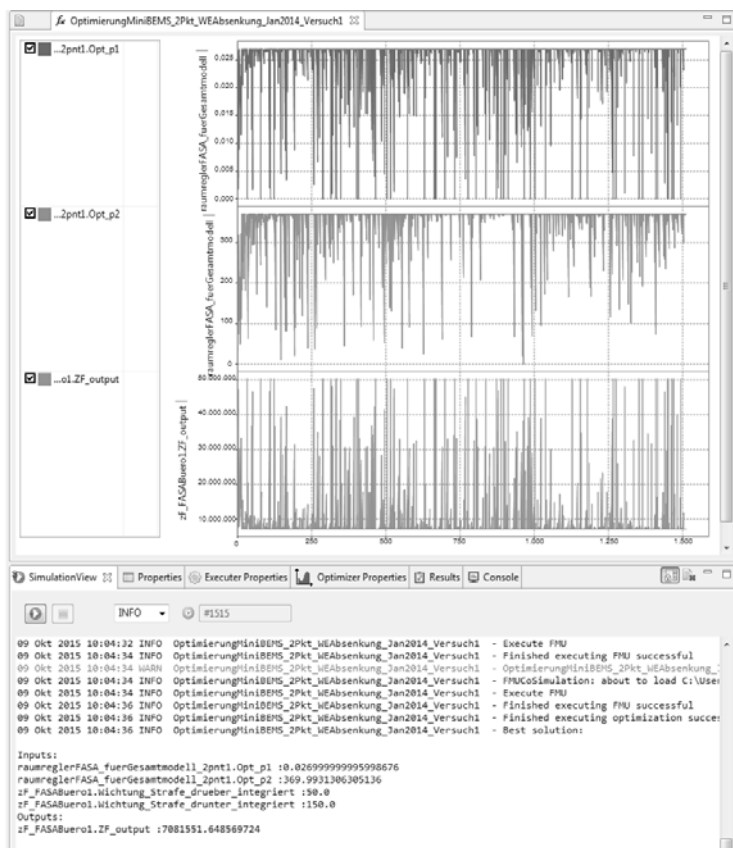


Figure 10: Optimization tool from EAS.

To minimize the energy consumption as well as assure comfort conditions the target function (7) was applied.

$$TF = E + p_{below} + p_{above} \quad (7)$$

In (7) E denotes the energy consumption of the floor heating system, p_{below} and p_{above} are penalty terms that penalize the target function TF when it is too cold or too warm in the room.

As mentioned the particle swarm algorithm was used for the optimization. Figure 10 shows a screenshot of the GUI of the optimization tool developed by Fraunhofer IIS EAS.

For the optimization the energy manager was connected to a room model to a complete optimization model which was exported as an FMU and connected to the optimizer via the FMI interface.

The optimized parameters are:

$$\begin{aligned} opt_p1 &= 0.027 \\ opt_p2 &= 370 \end{aligned}$$

Using these optimization parameters in the simulation of the complete model (energy manager connected to the room model) the energy consumption decreases from 120,7 kWh to 108,2 kWh while the comfort conditions are kept.

Figure 11 compares the simulated room temperature before and after the optimization.

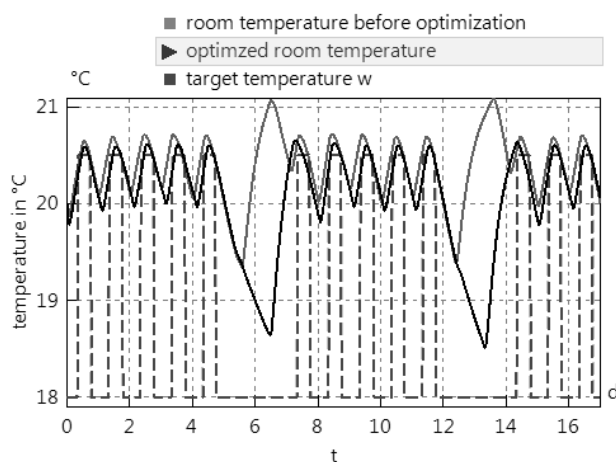


Figure 11: Room temperatures before and after optimization.

3.2 Energy manager for energy generating

As described in section 1, two differently detailed buildings models are available. Those models include, besides the stove, a heat pump which is not controlled by the energy manager.

When simulating the more detailed model including the developed energy manager for the first six months of the year using an arbitrarily chosen set of adjusting parameters, the energy supplied by stove and heat pump is calculated (second line in Table 2).

Unfortunately, this simulations takes around 5 hours (used hardware: Windows 7 Enterprise 64bit, Intel® Core™ i7-4600U CPU @ 2.7 GHz, 8 GB RAM). Since this is far too much for the application of optimization methods, the less detailed model is used since its simulation time is around 5 to 10 minutes.

The results of this simulation (third line of Table 2) show that the energy consumption of the stove and the heat pump is higher compared to the simulation of the more detailed model. Nevertheless, the less detailed building model is an approximation of the detailed model which describes the building behavior in principle correctly. Since its simulation time is much shorter, it is used for the optimization of the energy manager.

A further acceleration was reached by defining the adjusting parameters c_0 and c_2 to be of the type integer during optimization instead of real. The following intervals for the adjusting parameters were defined heuristically:

$$\begin{aligned} 0 \leq c_0 \leq 30, & & 0 \leq c_1 \leq 2 \\ 0 \leq c_2 \leq 20 & & 0 \leq c_3 \leq 1 \end{aligned}$$

The objective function is the sum of energy supplied by the heat pump as well as by the stove. Furthermore, penalty terms are added to ensure some temperature restrictions at the buffer.

$$C = E_{stove} + E_{heat\ pump} + penalty \quad (8)$$

Most important is a low energy consumption of the heat pump since it uses electric power that is expensive compared to the wood for the stove. The optimization results for the parameters and energy consumptions are shown in line 4 in Table 2. It can be seen, that the optimized adjusting parameters lead to a lower energy consumption compared to the simulation of the less detailed model with arbitrary parameters (line 3 of Table 2).

The optimized adjusting parameters are now used to simulate the original, more detailed model (line 5 in Table 2) and the results show that the optimized parameters of the less detailed model also lead to better results in the more detailed model (second line in Table 2). This gives the conclusion, that the model simplifications that led from the more detailed model to the less detailed one were allowable.

	c_0	c_1	c_2	c_3	Energy stove in kWh	Energy heat pump in kWh
detailed model	30	1	18	0.5	115	0
less detailed model	30	1	18	0.5	425	1421
less detailed model optimized	30	1.97	20	1	310	626
more detailed models with optimized parameters	30	1.97	20	1	94	0

Table 2: Optimization studies.

As could be seen during the optimization work, the performance of one simulation run can be quite bad since building models naturally are relatively extensive. Therefore, it is often too time consuming to finish optimization runs within reasonable time.

To overcome these difficulties, several strategies can be applied:

- **Take simplified models instead of accurate ones.**

It is possible to over-simplify models. Therefore, it needs to be investigated, how simple a model can be to generate still reasonable optimization results. At least the following consideration is useful: If simplified models are used for optimization, the parameters obtained from optimization should be applied to the more accurate model. Such a verification simulation should prove that the parameters still produce good results.

- **Reduce the number of parameters during optimization.**

A low number of parameters is advantageous for optimization runs. Parameters which have not a great influence on the optimization result can be identified by performing a sensitivity analysis.

- **Chose suitable time intervals for the parameter optimization.**

Often some states become active within dedicated time intervals only. Therefore, it is sometimes possible to optimize groups of parameters separately within shorter time intervals. This improves the performance. E.g. typically parameters which influence heating devices should not be optimized within summer months.

- **Do not simulate unreliable parameter values.**

Sometimes, the optimization algorithm choses unreliable as well as not realistic parameter constellations, which cause bad performant simulations or simulation crashes. Such obviously bad parameter constellations should be selected before simulation. This selection can be included into the simulation model.

- **Use parallelization.**

Both optimization as well as simulation can be accelerated by parallelization.

- **Choose good process parameters of the optimization method.**

The optimization method can be adapted by suitable optimization process parameters. This influences the performance of optimization drastically.

4 Summary

During the enerMAT project different BEMS respectively energy managers were developed. Since enerMAT follows a model based approach, the modelling of the FASA office building (two differently detailed models) was introduced. Additionally the development of two energy managers for the FASA office building was presented in this paper.


The energy manager for the energy generating part is developed as a statechart with adjusting parameters that are optimized to reach minimal energy consumption.

The energy manager for the heat consuming part was developed by process identification of the heating and cooling process using the example of one room. A bottleneck in this process is the often bad simulation and optimization performance. Therefore some strategies are discussed to improve the performance.

The investigation shows that a simple control algorithm can be adapted to a special room by optimization if suitable optimization parameters are introduced. The better the control algorithm is the less optimization parameters will change when other test cases (e.g. ambient conditions) are applied. This way the robustness as well as the quality of a control algorithm can be checked.

The potential of this approach is by far not exhausted yet: The optimization can comprise both adjusting parameters of the energy manager as well as constructing parameters of the building. Thus more common target functions are possible. Other investigations are necessary to support the development of the statechart.

Acknowledgement

The research is funded by	<p>Gefördert durch:</p>  <p>Bundesministerium für Wirtschaft und Technologie</p> <p>aufgrund eines Beschlusses des Deutschen Bundestages</p> <p>Förderkennzeichen: 03ET1084</p>
---------------------------	---

References

- [1] EU Energy and transport in figures, statistical pocket book 2007/2008
- [2] Clauss C, Haufe J, Blochwitz T et al.. Model Based Optimization of Building Control Systems. *Proceedings of the ASIM-Workshop STS/GMMS 2014*; 2014 Feb 20-21; Reutlingen-Rommelsbach, Germany; 2014, P. 131-138.
- [3] Website of the Austrian company "Technische Alternative" : <http://www.ta.co.at/de/produkte/uvr1611/frei-programmierbare-regelung-uvr1611.html>
- [4] Unger R, Schwan T, Mikoleit B et. al.. „Green Building“ – Modelling renewable building energy systems and electric mobility concepts using Modelica. *Proceedings of the 9th International Modelica Conference*; 2012 Sep 3-5; Munich, Germany; 2012, P. 897-906.
- [5] www.Modelica.org
- [6] Doukas H, Patlitzianas KD, Iatropoulos K et al.. Intelligent building energy management system using rule sets. *Building and Environment*; 2007; 42(10):3562-3569.
- [7] Kalogirou SA. Applications of artificial neural-networks for energy systems. *Applied Energy*; 2000; 67:17-35.
- [8] Kolokotsa D, Stavrakakis GS, Kalaitzakis K et al.. Generic algorithms optimized fuzzy controller for the indoor environmental management in buildings implemented using PLC and local operating networks. *Engineering Applications of Artificial Intelligence*; 2002; 15(5):417-428.
- [9] Castilla M, Álvarez JD, Berenguel M et al.. A comparison of thermal comfort predictive control strategies. *Energy and Buildings*; 2011;43:2737-2746.
- [10] Rosselló-Busquet A, Brewka LJ, Soler J et. al.. OWL Ontologies and SWRL Rules Applied to Energy Management. UKSim 13th International Conference on Modelling and Simulation; Emmanuel College Cambridge, United Kingdom; 2011, P. 446-450.
- [11] Byun J, Park S. Development of a Self-adapting Intelligent System for Building Energy Saving and Context-aware Smart Services. *IEEE Transactions on Consumer Electronics*; 2011; 57(1).

Implementation of Hybrid Systems Described by DEV&DESS in the QSS Based Simulator PowerDEVS

Franz Preyser^{1*}, Irene Hafner², Matthias Rößler¹

¹Department of Analysis and Scientific Computing, Vienna University of Technology, Wiedner Hauptstraße 8-10, 1040 Vienna, Austria; *franz.preyser@tuwien.ac.at

²Simulation Services, dwh GmbH, Neustiftgasse 57-59, 1070 Vienna, Austria

SNE Simulation Notes Europe SNE 25(2), 2015, 109-116
 DOI: 10.11128/sne.25.tn.10298
 Received: August 10, 2015 (Selected ASIM STS 2015 Postconf. Publ.); Accepted: August 15, 2015;

Abstract. In this article, a method for implementing hybrid models, formulated as DEV&DESS, in the QSS based simulator PowerDEVS is presented. PowerDEVS is actually a pure DEVS simulator. However, it is specialised for simulating continuous Models (DESS) using QSS (Quantized State System) to translate them into discrete event models (DEVS). Therefore, it is perfectly suited for the simulation of hybrid models (DEV&DESS). When designing and simulating coupled DEVS models though, very soon a lot of difficulties occur caused by concurrent events and feedback loops. Hence, first some concepts are introduced of how to implement an atomic DEVS in a way that avoids those difficulties. Afterwards, an approach of how to implement an atomic DEV&DESS is introduced which makes use of those concepts.

Introduction

The factor ‘costs for energy consumption’ is gaining more and more importance in terms of production process optimization. Since most of the time energy intense processes in a production line are of continuous nature a demand is rising for being able to formulate those continuous aspects in addition to the usual discrete logistical aspects in one simulatable model [1]. See [2] for preceding work on this issue.

Herbert Praehofer introduced DEV&DESS (Discrete Event & Differential Equation System Specification) [3] for the formal description of hybrid systems. DEV&DESS is based on the two formalisms DEVS and DESS invented by Bernard Zeigler [4]

for the description of discrete event systems and continuous systems, respectively.

However, as digital computers work in a purely discrete way, DESS models and therefore also DEV&DESS models have to be discretised somehow before they can be simulated on a digital computer. The usual method is to apply an ODE solver onto the differential equations describing the continuous model. A rather new alternative is called QSS (Quantized State System) [5] which is already mentioned in [4] as it transforms the continuous model (DESS) into a discrete event one (DEVS). Ernesto Kofman introduced a set of advancements to QSS [6],[7],[8],[9], [10] which have been implemented in PowerDEVS [11]. PowerDEVS is a DEVS simulator that supports graphical block orientated model description comparable to Simulink or Dymola.

Although PowerDEVS is specially designed for implementing continuous models in an discrete event manner, so far there has been no way to directly implement a DEV&DESS. A formal method of how to embed DEV&DESS in DEVS is presented in [12]. Based on this work, a generic PowerDEVS DEV&DESS block is developed.

However, creating and simulating coupled DEVS models turns out to be quite difficult. This mainly is owed to the various numbers of possible scenarios of concurrent events that may occur during simulation and that have to be considered when defining the DEVS of each single block involved. To relieve the modeller of these difficulties, a new PowerDEVS *Atomic DEVS* block is introduced, based on the DEVS extension *Parallel DEVS* (P-DEVS, see [13]). The thereby developed methods for concurrency treatment are then also utilized for the mentioned generic DEV&DESS block, called *Atomic DEV&DESS*.

1 Theoretical Background

1.1 DEVS

DEVS denotes a formalism for describing systems that allow changes only at discrete points in time called *events*. An atomic DEVS is specified by the following 7-tuple.

$$\langle X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta \rangle$$

where

- $X \dots$ set of possible inputs (e.g. \mathbb{R}^n)
- $Y \dots$ set of possible outputs (e.g. $\mathbb{R}^+ \times \mathbb{N} \times \mathbb{R}^m$)
- $S \dots$ set of possible states (=state space)
- $Q = \{(s, e) | s \in S, e \in [0, ta(s)]\}$
- $\delta_{ext} : Q \times X \rightarrow S \dots$ external state transition function
- $\delta_{int} : S \rightarrow S \dots$ internal state transition function
- $\lambda : S \rightarrow Y \dots$ output function
- $ta : S \rightarrow \mathbb{R}_0^+ \cup \infty \dots$ time advance function

Figure 1 illustrates a DEVS graphically. It basically consists of an inner state s that can be altered by the external and internal state transition functions which are evaluated each time an external or an internal event occurs. External events are triggered by arriving input messages x , whereas internal events are triggered when the current state s has not changed for the duration of $ta(s)$. On each internal event, right before δ_{int} is called, the output function λ is evaluated which may result in an output message y .

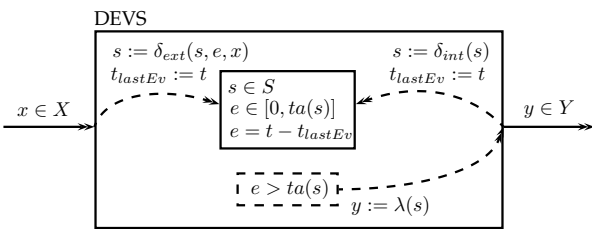


Figure 1: Graphical illustration of an atomic DEVS.

As an atomic DEVS has input and output ports, it can be coupled with other atomic DEVS resulting in a *coupled DEVS* whose behaviour again can be described by an atomic DEVS (see closure under coupling property in [4]). Figure 2 illustrates some possible coupling schemes of atomic and coupled DEVS.

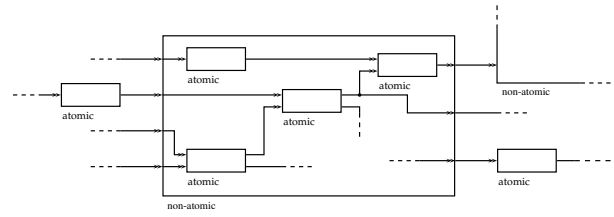


Figure 2: Coupling scheme of DEVS blocks.

Two types of DEVS can be distinguished:

Definition 1.1 Mealy Type DEVS

A DEVS is called Mealy Type DEVS or of Type Mealy, if there exists an internal state s and an external input x in such a way that $ta(\delta_{ext}(s, x)) = 0$. Thus, a model described by a mealy type DEVS may produce an output as immediate response to an input.

Definition 1.2 Moore Type DEVS

A DEVS is called Moore Type DEVS or of Type Moore, if it is not of type mealy.

1.2 DEV&DESS

DEV&DESS formalism is a composition of DEVS and DESS and therefore capable of describing hybrid systems. An atomic DEV&DESS can be described by the following 11-tuple:

$$\langle X^{discr}, X^{cont}, Y^{discr}, Y^{cont}, S, \delta_{ext}, C_{int}, \delta_{int}, \lambda^{discr}, f, \lambda^{cont} \rangle$$

where

- $X^{discr}, Y^{discr} \dots$ set of discrete inputs and outputs
- $X^{cont}, Y^{cont} \dots$ set of continuous inputs and outputs
- $S = S^{discr} \times S^{cont} \dots$ set of states (=state space)
- $Q = \{(s^{discr}, s^{cont}, e) | s^{discr} \in S^{discr}, s^{cont} \in S^{cont}, e \in \mathbb{R}_0^+\}$
- $\delta_{ext} : Q \times X^{cont} \times X^{discr} \rightarrow S \dots$ external transition fct.
- $\delta_{int} : Q \times X^{cont} \rightarrow S \dots$ internal transition function
- $\lambda^{discr} : Q \times X^{cont} \rightarrow Y^{discr} \dots$ discrete output function
- $\lambda^{cont} : Q \times X^{cont} \rightarrow Y^{cont} \dots$ continuous output function
- $f : Q \times X^{cont} \rightarrow S^{cont} \dots$ rate of change function
- $C_{int} : Q \times X^{cont} \rightarrow \{true, false\} \dots$ event condition fct.

As it can be seen, apart from ta , each component of an atomic DEVS is recurring in an atomic DEV&DESS. Additionally there are the right side of the ODE f

and the continuous output function λ^{cont} describing the DESS part. The state event condition function C_{int} though, is new. It is responsible for triggering internal events in the DEV&DESS and therefore replaces ta . However, C_{int} does not only trigger time events, but also state events, i.e. events caused by the internal state q reaching some specific value. Nevertheless, both time and state events result in an execution of δ_{int} .

Figure 3 shows a graphical illustration of an atomic DEV&DESS. As well as DEVS also DEV&DESS is closed under coupling. However, continuous output ports cannot be coupled arbitrary to discrete input ports, but only if their output value is guaranteed to be piecewise constant. See [4] for more details about that and about DEVS and DEV&DESS in general.

2 DEVS Simulation in PowerDEVS

In PowerDEVS a DEVS of an atomic block consists of the specification of the six C++ functions: $init(t, parameters)$, $ta(t)$, $dint(t)$, $dext(x, t)$, $lambda(t)$ and $exit()$. These functions are member functions of a C++ class describing the block. Thus, state variables, block parameters and auxiliary variables are defined as member variables of that class making them accessible in all the member functions.

These member functions are called by the PowerDEVS simulation engine whenever their execution is necessary to calculate the models dynamic behaviour. The function $init(t, parameters)$ is called right before the actual simulation is started and can be used to initialise the system's state and to read block parameter values that have been entered by the modeller using PowerDEVS graphical user interface and the block's *Parameters Dialogue*. The function $exit()$ is called after the simulation is finished and thus, can be used for example to free allocated memory.

The other methods represent the corresponding functions of the DEVS formalism. In case of an internal event they are executed in the following order:

1. call of $lambda(t)$
2. $e = t - t_l$
3. call of $dint(t)$
4. $t_l = t$
5. call of $ta(t): t_n = t + ta(t)$

where t_l denotes the time of the last event, and t_n the time of the next event in the corresponding DEVS.

In case of an external event they are executed in the following order:

1. $e = t - t_l$
2. call of $dext(x, t)$
3. $t_l = t$
4. call of $ta(t): t_n = t + ta(t)$

In coupled DEVS the so-called *Select* function, a kind of priority ranking of the involved blocks, selects which DEVS is allowed to execute its internal transition function first, when several of them are scheduled simultaneously. In PowerDEVS the *Select* function is implemented as a list in which all blocks occurring in a coupling are sorted descending according their priority in case of concurrent internal events. However, if a block produces an output message, the external transition function at the receiving block is always executed immediately, independent of its priority.

3 Problems with DEVS and Solution Approaches

3.1 Problem identification

The definition of an atomic DEVS behaving exactly as intended in every possible situation of concurrent input messages at different input ports turns out to be quite challenging. The *Select* function regulating the resolution of such concurrencies is part of the coupling but nevertheless influences the behaviour of an atomic DEVS in such situations and therefore, it is hard to consider when formulating the atomic DEVS. For example, if an atomic DEVS with the current internal state s receives the input messages x_1 at its input port one and x_2 at its input port two at the same instant of simulation time, it depends on the *Select* function whether its new state calculates as $\delta_{ext}(\delta_{ext}(s, x_2), x_1)$ or as $\delta_{ext}(\delta_{ext}(s, x_1), x_2)$. Even more, if the block is of type Mealy, it may depend on the *Select* function if an output message is produced in reaction to the input messages or not.

Therefore, to define a DEVS in a rigorous way, it is necessary to consider each possible set of concurrent input messages and moreover, every possible treatment order.

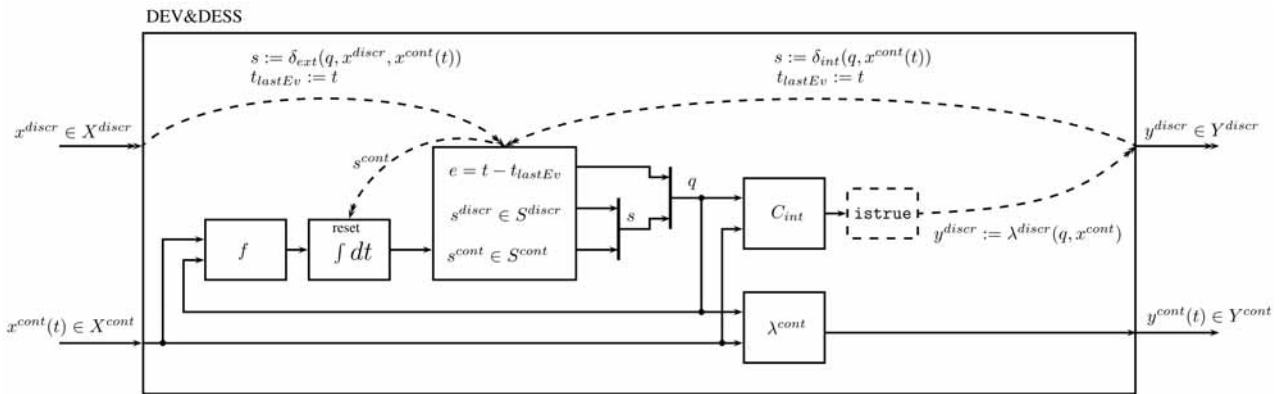


Figure 3: Graphical illustration of an atomic DEV&DESS.

3.2 Parallel DEVS approach

The DEVS extension *Parallel DEVS* counters the problems mentioned in section 3.1 by collecting all input messages in so called *bags* before treating them all at once in one single call of δ_{ext} . If at the same time also an internal event is triggered, a so called *confluent transition function* $\delta_{conf} : Q \times X \rightarrow S$ is applied instead of δ_{ext} .

The idea is to first calculate λ of each immanent block, i.e. of each block experiencing an internal transition at the current simulation time, before calculating δ_{int} or δ_{conf} of any of them. However, due to blocks of type Mealy and due to feedbacks in the coupling it may be necessary to recalculate λ if the set of arrived input messages of the corresponding block has changed after its former calculation. Therefore, the determination of a stable set of input messages for each block is an iterative process. It even is possible that finally the stable set of input messages is empty although having not been empty in former iteration which makes it necessary to undo the state changes that may have been accomplished by a call of δ_{ext} .

If a coupled model contains an algebraic loop, it may not be possible to determine a stable set of input messages for each block in a finite number of iterations. Such models are called *illegitimate*.

3.3 DEVS approach in PowerDEVS

As in Parallel DEVS the execution of λ is decoupled from the succeeding execution of the δ function, P-DEVS works with a simulation engine different to the one used for DEVS. PowerDEVS does not support Parallel DEVS simulation. Nevertheless, it is possible to implement P-DEVS functionality in PowerDEVS.

For this purpose, three mechanisms have to be installed. The first one addresses the gathering of concurrent input messages in a set \mathbf{x} (bold letters denote sets). In order to do that the internal state of the DEVS is extended by an input buffer representing \mathbf{x} and the external transition function only is allowed to change the input buffer (by storing arriving messages with their arrival time in it). The actual state change caused by the external event is shifted into the internal transition which is executed every time the input buffer has been modified. Thus, it is made sure that all input messages coming from blocks with higher priority are gathered in \mathbf{x} before they are treated. However, due to feedback couplings it cannot be avoided that some input messages origin from blocks with lower priority. This problem is addressed by the second mechanism.

The second mechanism consists of a backing up of the state s of the DEVS every time the first event at the current simulation time is triggered. In PowerDEVS this can be identified by $t_{1} < t$. The δ functions (δ_{int} , δ_{ext} , δ_{conf}) are then using the backup s_{old} instead of s to calculate a new state. Therefore, if \mathbf{x} is changed after one of the δ functions has already been evaluated it simply is re-evaluated: $s = \delta(s_{old}, \mathbf{x})$. As every calculation of δ is preceded by a calculation of λ though, it may occur that formerly output messages have been produced at output ports where finally no output messages are to be sent. However, these output messages have already altered the set \mathbf{x} of its receiving blocks. These alterations have to be withdrawn somehow. This is what the third mechanism is dedicated to.

Like \mathbf{x} also the set of output messages \mathbf{y} is iteratively changing and hopefully finally stabilising. Thus, also for the output messages an output buffer is installed as part of the state of the DEVS. Each calculated output

message is stored in it at the corresponding output port with its time of last change and with an 'already sent - flag'. So if λ is recalculated and there exists an entry in the output buffer that has been already sent at the current simulation time but is not marked to be resent (as it is not included in the result of λ anymore) a *retrieve message* is sent instead informing the receiving block to ignore the formerly received message and to recalculate its own λ and δ function.

4 Atomic PDEVs Block

The created *Atomic PDEVs* PowerDEVS library block is intended to overcome the problems mentioned in section 3.1 by implementing the three mechanisms discussed in section 3.3. In the following, its working principle is expressed in form of a description of the content of the C++ functions corresponding to δ_{ext} , ta , λ , and δ_{int} . Of course, a complete description of each detail that need to be considered when programming the *Atomic DEVs* block would go beyond the scope of this article.

1. $dext(x, t)$:
 - If $t_l < t$ set $s_{old} = s$, $\sigma_n = \sigma - e$, $\sigma_{n,old} = \sigma_n$ and $flag = 'n'$ but if additionally $\sigma_n = 0$ set $flag = 'i'$.
 - If x is a retrieve message, remove the corresponding entry from \mathbf{x} , set $\sigma = 0$ and if $\mathbf{x} = \emptyset$ set $flag = 'n'$.
Else, if x differs from the last reception at the same port in value or in arrival time, modify \mathbf{x} accordingly and set $\sigma = 0$ and update $flag: 'n' \mapsto 'e', 'i' \mapsto 'c'$.
2. $ta(t)$:
Return σ .
3. $lambda(t)$:
 - If $t_l < t$ set $s_{old} = s$ and $flag = 'i'$
 - If there is no pending output message, calculate $\mathbf{y} = \lambda(s_{old}, \mathbf{x})$, and add retrieve messages to \mathbf{y} .
 - If there are pending output messages left in \mathbf{y} , output one of them.
4. $dint(t)$:
If there are no pending output messages left and no new inputs arrived during outputting \mathbf{y} :
 - if $flag = 'i'$ calc. $[s, \sigma_n] = \delta_{int}(s_{old})$.
 - if $flag = 'e'$ calc. $[s, \sigma_n] = \delta_{ext}(s_{old}, \mathbf{x})$.

- if $flag = 'c'$ calc. $[s, \sigma_n] = \delta_{conf}(s_{old}, \mathbf{x})$.
- if $flag = 'n'$ set $[s, \sigma_n] = [s_{old}, \sigma_{n,old}]$.
- set $\sigma = \sigma_n$.

The variable $flag$ is used to identify the type of event and therefore, which particular δ function is to be used for calculating the new state. The message retrieving mechanism though, only works if all blocks involved in a coupling implement it.

5 Atomic DEV&DESS Block

Based on the Atomic PDEVs block, an *Atomic DEV&DESS* PowerDEVS library block is developed.

5.1 Structure

The structure of the DEV&DESS embedded in PowerDEVS is depicted in Figure 4. The idea is to divide a DEV&DESS into a continuous (DESS) part that can be implemented graphically as block diagram and into a discrete part that can be implemented as atomic PDEVs block. For this purpose the function C_{int} is also divided into two parts: one for detecting state events (C_{int}^{se}) and one for scheduling time events (C_{int}^{te}). The first one is a component of the continuous part and the second one is included in the atomic PDEVs definition.

The continuous part consists of λ^{cont} , f , an integrator \int , and of C_{int} . The discrete part, responsible for the implementation of δ_{ext} , δ_{int} , λ^{discr} , and C_{int}^{te} is realized as one single DEVs referred to as *main block*. The priority list of the coupling in Figure 4 is as follows: f , \int , C_{int}^{se} , *main block*, λ^{cont} .

5.2 QSS signals

All continuous signals in DEV&DESS are described as QSS signals in PowerDEVS. That is, they are described as piecewise polynomial functions with jump discontinuities at the merging points of two polynomials which are smaller than a requested constant called *quantum* q . The polynomials represent the Taylor polynomials of the continuous signal with the expansion point at the discontinuities. Every time the difference between the current Taylor polynomial and the continuous signal or between Taylor polynomial and a Taylor approximation of higher order becomes equal to q , the expansion point is advanced in time and the coefficients of the new polynomial are sent as DEVs message. In this way, a continuous signal can be described in a discrete event man-

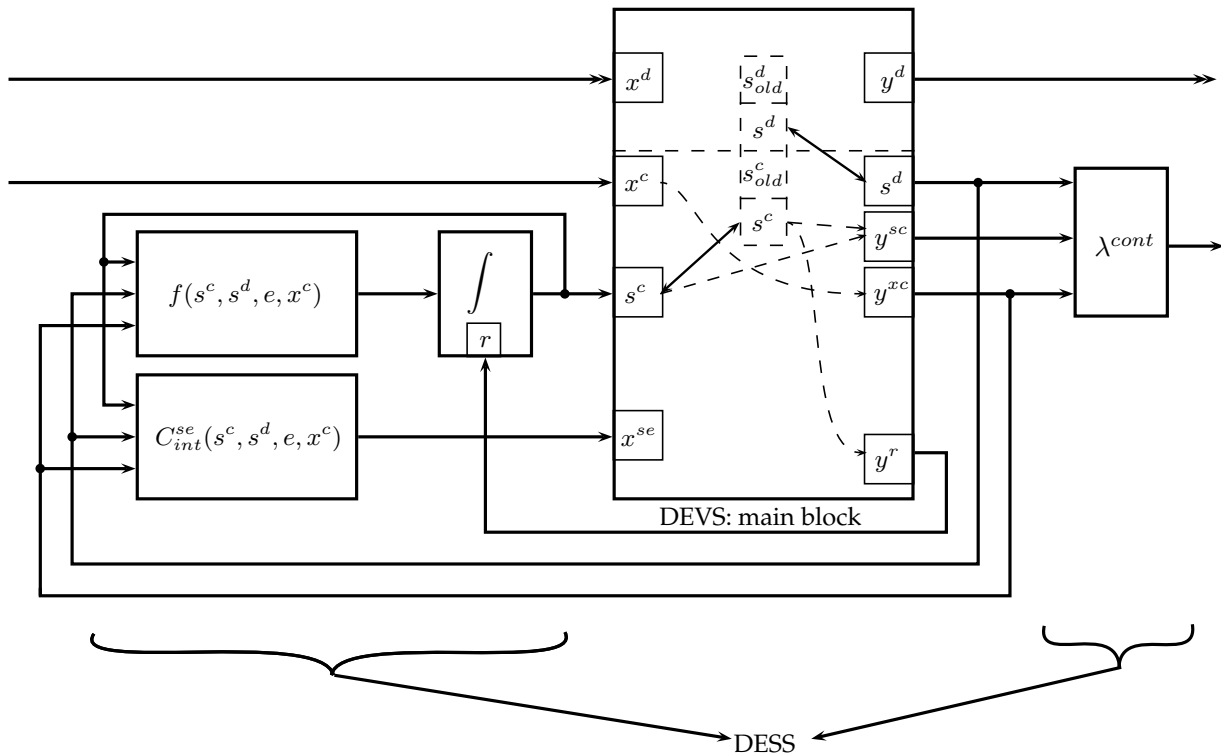


Figure 4: Graphical illustration of the construction of a DEV&DESS in PowerDEVS.

ner. As in DESS piecewise continuous signals are allowed, jump discontinuities bigger than q may appear as well. Anyway, they are not caused by the signal discretisation but have already existed before.

5.3 State Events

Using QSS, state events, triggered by the continuous state reaching a specific value, simply can be calculated as the points in time when the current polynomial signal representation reaches the specified value. Thus, state events are transformed into time events.

All the signal lines in Figure 4 may transmit vectorial signals. However, in PowerDEVS vectorial signals are not sent at once but index by index resulting in temporally non-valid signals during the time after the first changed index is sent and before the last changed index is sent. This in turn may lead to state events wrongly indicated by C_{int}^{se} .

Further there is the case in which a state transition in the main block leads to a state that immediately triggers a state event. This case is declared to be prohibited. That is, the DEV&DESS is not allowed to define state transitions leading immediately to a state event.

5.4 Main Block

As the main block calculates δ_{ext} and δ_{int} it has to administer the entire state of the system consisting of a discrete part s^d and of a continuous part s^c . However, in between two state transitions in the main block s^c may change due to Taylor polynomial expansion point advancement triggered by the integrator. Therefore, the third input port of the main block is coupled to the integrator's output port. The input buffer of that port is simultaneously used as storage for the continuous state s^c itself. The same holds for s^d and the output buffer of the second output port. The fourth input port of the main block receives the result of the calculation of C_{int}^{se} and thus, input messages at this port trigger internal transitions.

f , C_{int}^{se} , and λ^{cont} depend on the continuous input signal x^c . However, they are not coupled directly to it but indirectly through the main block. This is to not forward each single, maybe even only temporary index change in x^c immediately to the continuous part. Instead the whole new x^c is forwarded just before the main block calculates one of the δ functions as it has to evaluate C_{int}^{se} before to being able to decide which one.

In the following, the working principle of the main block is expressed in form of a description of the content of the C++ functions corresponding to δ_{ext} , ta , λ , and δ_{int} .

1. `dext(x, t)` :

If $t_1 < t$ set $s_{old} = s$, $\sigma_n = \sigma - e$, $\sigma_{n,old} = \sigma_n$,
 phase='g' and flag='n',
 if additionally $\sigma_n = 0$ set flag='i'.

If input at port:

- x^d : add/remove x to/from \mathbf{x}^d .
 update flag:
 if $\mathbf{x} = \emptyset$ 'e' \mapsto 'n', 'c' \mapsto 'i'
 otherwise, 'n' \mapsto 'e', 'i' \mapsto 'c'
- x^c : add/remove x to/from \mathbf{x}^c .
- s^c : add change in s^c to \mathbf{y}^{sc} .
 if phase='g', apply change in s^c to s_{old}^c
- x^{se} : add x to \mathbf{x}^{se}

If any input buffer changed, set $\sigma = 0$.

2. `ta(t)` :

Return σ .

3. `lambda(t)` :

If $t_1 < t$ set $s_{old} = s$, $\sigma_n = \sigma - e$, $\sigma_{n,old} = \sigma_n$,
 phase='l' and flag='i'

if phase=

'g' Set phase='c'.

'c' If \mathbf{x}^c changed, or $s \neq s_{old}$, forward \mathbf{x}^c to
 \mathbf{y}^{xc} , set $s^d = s_{old}^d$ and $\mathbf{y}^r = s_{old}^c$.
 Further set phase='r'.
 Else set phase='C'.

'r' If there is a pending output left in \mathbf{y} ,
 set $\mathbf{x}^{se} = \emptyset$ and output next message.
 Else, set phase='C'.

'C' If $x^{se} \neq \emptyset$, update flag: 'n' \mapsto 'I',
 'e' \mapsto 'C'
 phase='l'

'l' if flag='i' | 'I':

calc. $[s^d, s^c, \sigma_n] = \delta_{int}(s_{old}^d, s_{old}^c, \mathbf{x}^c)$,
 and $\mathbf{y}^d = \lambda^{discr}(s_{old}^d, s_{old}^c, e, \mathbf{x}^c)$.

if flag='e':

calc. $[s^d, s^c, \sigma_n] = \delta_{ext}(s_{old}^d, s_{old}^c, e, \mathbf{x}^c, \mathbf{x}^d)$,
 and $\mathbf{y}^d = \lambda^{discr}(s_{old}^d, s_{old}^c, e, \mathbf{x}^c)$.

if flag='c' | 'C':

calc. $[s^d, s^c, \sigma_n] = \delta_{conf}(s_{old}^d, s_{old}^c, e, \mathbf{x}^c, \mathbf{x}^d)$,
 and $\mathbf{y}^d = \lambda^{discr}(s_{old}^d, s_{old}^c, e, \mathbf{x}^c)$.

Add retrieve messages to \mathbf{y}^d .

Add each change in s^c to \mathbf{y}^r .

Set phase='o'.

'o' if unsent element of \mathbf{y}^r left, send it.

Else, if s^c changed, update \mathbf{y}^{sc} .

Else, if unsent element of \mathbf{y}^d left, send it.

Else, if unsent change of s^d left, send it.

Else, if unsent element of \mathbf{y}^{sc} left, send it.

4. `dint(t)` :

If now pending output is left and there are no
 new input messages at input ports x^d and x^c ,
 set $\sigma = \sigma_n$.

As the new state is part of the output of the main block the order in which λ and δ are calculated is reversed here. This is why δ is now calculated in the C++ function `lambda` instead of in `dint`.

Since each time before δ and λ are calculated, C_{int}^{se} has to be evaluated, there are several different phases in `lambda` to be distinguished. This is what the variable `phase` is for.

Although the description above is quite elaborate, it still does not cover every detail of the complete source code of the *Atomic DEV&DESS* block. However, the basic principles are included.

6 Conclusion

With the ever growing computational power of modern computers also the possibilities to simulate more and more extensive and complex models increase. However, when including more and more details into a model, very soon a point is reached where pure discrete or pure continuous models do not suffice anymore. Production process models which include energy consumption behaviour provide an example for this trend. As a consequence the formulation and simulation of hybrid models is demanded. Concerning the formulation, DEV&DESS seems to be quite powerful. So far though, it lacks a simulator able to rigorously implement and simulate a DEV&DESS. An approach to implement DEV&DESS in PowerDEVS has been demonstrated in this paper. As the correct definition of coupled DEVS models is quite challenging, additionally a way of how to implement models in PowerDEVS similar to Parallel DEVS has been introduced. However, profound theoretical investigations and a formal proof of the correctness of the presented approaches are works that still need to be done.

References

- [1] Popper N, Hafner I, Rössler M, Preyser F, Heinzl B, Smolek P, Leobner I. A General Concept for Description of Production Plants with a Concept of Cubes. *Simulation Notes Europe*. 2014;24(2):105–114.
- [2] Schmidt A, Pawletta T. Hybride Modellierung fertigungstechnischer Prozessketten mit Energieaspekten in einer ereignisdiskreten Simulationsumgebung. In: *ASIM 2014 – 22. Symposium Simulationstechnik, Berlin, 03.-05.09.2014*. ARGESIM/ASIM. 2014; pp. 109–116.
- [3] Praehofer H. Systems Theoretic Formalisms for Combined Discrete-Continuous System Simulation. *International Journal of General Systems*. 1991; 19(3):219–240.
- [4] Zeigler B, Praehofer H, Kim T. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press. 2000.
- [5] Kofman E, Junco S. Quantized State Systems. A DEVS Approach for Continuous System Simulation. *Transactions of SCS*. 2001;18(3):123–132.
- [6] Kofman E. A Second Order Approximation for DEVS Simulation of Continuous Systems. *Simulation: Transactions of the Society for Modeling and Simulation International*. 2002;78(2):76–89.
- [7] Kofman E. Quantized-State Control. A Method for Discrete Event Control of Continuous Systems. *Latin American Applied Research*. 2003;33(4):399–406.
- [8] Kofman E. Discrete Event Simulation of Hybrid Systems. *SIAM Journal on Scientific Computing*. 2004; 25(5):1771–1797.
- [9] Kofman E. A Third Order Discrete Event Simulation Method for Continuous System Simulation. *Latin American Applied Research*. 2006;36(2):101–108.
- [10] Kofman E. Relative Error Control in Quantization Based Integration. *Latin American Applied Research*. 2009;39(3):231–238.
- [11] Bergero F, Kofman E. PowerDEVS. A Tool for Hybrid System Modeling and Real Time Simulation. *Simulation: Transactions of the Society for Modeling and Simulation International*. 2011;87(1–2):113–132.
- [12] Zeigler B. Embedding DEV&DESS in DEVS. In: *DEVS Integrative M&S Symposium (DEVS'06)*; .
- [13] Chow ACH, Zeigler BP. Parallel DEVS: a parallel, hierarchical, modular, modeling formalism. In: *Winter Simulation Conference'94*. 1994; pp. 716–722.

Domain-Specific Languages for Flexibly Experimenting with Stochastic Models

Danhua Peng^{*}, Tom Warnke, Adelinde M. Uhrmacher

Modeling and Simulation Group, Institute of Computer Science, University of Rostock, Albert-Einstein-Straße 22, 18059 Rostock, Germany; ^{*} *danhua.peng2@uni-rostock.de*

Simulation Notes Europe SNE 25(2), 2015, 117 - 122

DOI: 10.11128/sne.25.tn.10299

Received: August 10, 2015 (Selected ASIM STS 2015 Postconf. Publ.); Accepted: August 15, 2015;

Abstract. Developing a model for simulation is a difficult task, in which simulation experiments play a critical role. In modeling and simulation, domain specific languages are widely used for model description. More and more efforts have been put in facilitating simulation reproducibility in recent years. This motivates the use of domain specific languages as the means to express experiment specifications.

Domain specific languages can be used to specify different tasks of simulation experiments, such as experiment configuration, observation, analysis, and evaluation of experimental results. More importantly, they can serve to specify crucial observations from experiments regarding model behavior. Therefore, with a formal description of model behavior, an evaluation based on model checking techniques can also benefit from domain specific languages.

In this paper, we will first discuss how domain specific languages can be used to specify simulation experiments and illustrate it by using the domain specific language SESSL. We aim at dealing with stochastic models. Several problems arise in specifying simulation experiments with stochastic models, such as probability estimation, tolerating stochastic noises, and robustness measurement. Domain specific languages can help handling those problems.

Introduction

Building simulation models is a complex process, which is both an art and a science. To ensure that models are created at the appropriate level of abstraction and are valid regarding certain questions of interest, the design and execution of simulation experiments is essential. On the one hand, the reproducibility of simulation experiment results is or should be a basic requirement for model publication.

On the other hand, in the model development process, the model may be revised iteratively. After the model revision, it may be necessary to repeat the simulation experiments conducted with previous versions of model. Besides, when new models are built based on this model, those simulation experiments can provide useful information to assist experimentation with the new models as well [1].

Therefore, it is of significance to describe simulation experiments so that they can be easily reproduced. To enable this, an unambiguous, explicit experiment description is important. All the aspects that define the simulation experiment should be recorded completely and accurately, including the conditions and the results generated from the experiments.

The advantages of domain specific languages for model design are well-known. They enable domain experts to build models using the vocabulary of the domain, while hiding implementation details. For example, languages for cell biological models such as ML-Rules [2] adopt a rule-based modeling style that resembles biochemical reaction equations, whereas the object-oriented style of Modelica [3] can easily be mapped to components of technical systems.

However, domain specific languages can not only be used to create models, but also to support flexibly experimentation with models. In modeling and simulation, there is a trend that treats the experimentation process as a first class object, e.g., in [4] and [5], where several individual tasks can be distinguished in this process, such as configuration, data collection, analysis, and evaluation.

In this paper, we will first present the domain specific language SESSL (Simulation Experiment Specification via a Scala Layer) [6]. As our focus so far has been on experiments with stochastic models, we will discuss the problems and challenges in dealing with stochasticity and how they can be handled by extensions of SESSL.

1 Domain Specific Languages in Experiment Specification

1.1 Domain Specific Languages

A domain-specific language (DSL) is a programming language that is targeted specifically at an application domain, in contrast to a general purpose language. It contains syntax and semantics that represent the concept at the same level of abstraction that the application domain offers [7]. According to [8], a DSL is small and declarative, and offers expressive power focused on and usually restricted to a particular problem domain through appropriate notations and abstractions.

Typically, two types of domain specific languages are distinguished: internal (or embedded) DSL and external DSL. An embedded DSL is implemented based on a general-purpose programming language, i.e., the host language, as an embedding. It inherits the constructs of its host language and adds domain-specific primitives to provide the user a suitable modeling abstraction. However, its use requires typically some knowledge of the host language. The advantage of internal domain specific languages is that less implementation effort is required for designing. More importantly, they can easily be extended.

An external domain specific language, in contrast to internal domain specific languages, is developed as an independent language, which requires separate interpretation or compilation. They are designed ground-up; therefore their development has more freedom without constraints from the host language. In modeling and simulation, both types of domain specific languages are used [9].

1.2 Specifying simulation experiments

One goal of specifying simulation experiments is to allow reproducibility of the experiment and their exchange among different scientific groups. For that, one has to identify what kind of information is required to reproduce experiment results.

Several work exists on identifying requirements in describing simulation experiments, such as Minimum Information About a Simulation Experiment (MIASE) [10] and Minimum Simulation Reporting Requirements (MSRR) [11]. More detailed information can be found in [12].

These standards define guidelines in providing an accurate and complete description of simulation experiments. As identified by MIASE, to make the description of simulation experiments available to third parties, it must contain: the models to be simulated and their configuration parameters, the simulation configuration such as simulator to be used, the post-processing on the raw numerical results and the description of the final output results [10]. Simulation experiments can be interpreted as a process that comprises different tasks. In [5], six tasks are identified in a simulation experiment: specification, configuration, simulation, data collection, analysis and evaluation.

By combining the two perspectives above, we argue that domain specific languages, as being able to allow the reproducibility of simulation experiments, can be employed to support simulation experimentation on models from different aspects: model configuration, simulation configuration, experiment execution, observation, analysis, and evaluation of results. We will illustrate this with the domain specific language SESSL.

2 SESSL

SESSL is an embedded domain-specific language for simulation experiments [6]. It exploits the feature of its host language Scala [13], such as meta-programming, to allow flexible experiment set-ups. A SESSL specification can incorporate simulation algorithm, model parameters, simulation run time, parallel execution, stopping conditions, replication numbers, observation, result analysis a.s.o., as needed; however only the specification of the model file is mandatory while a default option is provided for the rest. The actual experiment is then performed with arbitrary simulation software that is controlled by SESSL based on a specific binding. Currently, a number of bindings to different simulation systems exist, such as the binding to the modeling and simulation framework JAMES II [14]; additional bindings can be added straightforwardly.

We illustrate the features of SESSL with an experiment specification as shown in Listing 1. In this example, a simulation experiment is specified based on JAMES II (line 2). This specification contains configuration of the model (line 4-6), configuration of the execution machinery (line 7-10), observation (line 11-12), evaluation of results (line 13-18) and execution (line 20).

```

1 import sessl._ // SESSL core
2 import sessl.james._ // JAMES II binding
3 val exp = new Experiment with Observation with ParallelExecution with Hypothesis {
4   model = "file-mrj:/./SimpleModel.mrj"
5   scan("a" <- range(100, 50, 200), "b" <- range(1, 4, 10), "c" <- range(0.01, 0.2, 1))
6   set("d" <- 10.0)
7   simulator = MLRulesTauLeaping()
8   stopCondition = AfterWallClockTime(seconds = 10) or AfterSimTime(1)
9   replications = 100
10  parallelThreads = -2
11  observe("x")
12  observeAt(range(0.0, 0.1, 10.0))
13  assume{
14    P(Peak("x", "peakHeight"), time >= 2 and time <= 4, "peakTime"),
15    E(Decrease("x", "decreaseAfterPeak"), end = 10, "afterPeak"),
16    Id("peakTime") STARTS Id("afterPeak"),
17    Id("peakHeight") >= Id("decreaseAfterPeak")
18  }
19 }
20 execute(exp)

```

Listing 1: SESSL specification of a simulation experiment based on a binding to JAMES II.

As SESSL can easily support experiment set-up and execution, other analysis such as optimization is provided as well and examples on this can be found in [6]. We moved our focus to the analysis of results. Model checking is a well-established verification technique to automatically analyze the dynamic behavior of models, based on formalizing model behavior with temporal logics, such as Linear Temporal Logic (LTL) [15]. To support this, we extended SESSL with an additional trait ‘Hypothesis’ in [1], which allows to specify behavior properties with LTL. Another language was proposed to describe the properties of trajectories in [16], and integrated into SESSL. As shown in Listing 1, a property is specified (line 13-18), which states that the model variable ‘x’ observed from experiments reaches a peak between time 2 and time 4, followed by a decrease which ends at time 10. This example shows how domain specific languages, like SESSL, are capable to support different tasks of simulation experiments. Meanwhile, with the explicit, declarative SESSL experiment specification, simulation experiments can be also reproduced.

3 Experimentation on Stochastic Models

In many areas such as systems biology, stochasticity plays an important role. Stochastic models, e.g., Continuous-Time Markov Chains (CTMC), provide a powerful means to model and to analyze the dynamics of the sys-

tem of interest. When conducting experimentation with stochastic models, certain problems need to be considered.

3.1 Probability estimation

Simulation experiments with a stochastic model require multiple replications to gain the confidence on experiment results. To analyze and evaluate the experiment results, a typical question arises: what is the probability that the model shows a certain behavior? Statistical model checking [17], which is a simulation-based verification technique, has been widely used to provide answers to this question. Several statistical model checking approaches exist, such as the Bayesian approach [18] and the Sequential Probability Ratio Test [19].

To support statistical model checking, we extended SESSL to allow the definition of probabilistic statements based on Continuous Stochastic Logic and hypothesis testing [1]. The property of model behavior can be specified in either LTL or the trajectory language proposed in [16]. Listing 2 specifies that with a probability of at least 0.8, the variable ‘x’ shall peak between time 2 and time 4 and afterwards decrease until time 10. Using hypothesis testing, the number of required simulation replications can be determined. A corresponding number of simulation trajectories are generated, against each of which the property specification is checked. Thus, it is possible to determine whether the model satisfies the specification with a probability greater than a given threshold.

```

1 assume(Probability >= 0.8){
2     P(Peak("x", "peakHeight"), time >= 2 and time <= 4, "peakTime"),
3     E(Decrease("x", "decreaseAfterPeak"), end = 10, "afterPeak"),
4     Id("peakTime") STARTS Id("afterPeak"),
5     Id("peakHeight") >= Id("decreaseAfterPeak")
6 }
    
```

Listing 2: A probability property specification in SESSL experiment: with a probability of at least 0.8, the number variable ‘x’ shall peak between time 2 and time 4 and afterwards decrease until time 10.

3.2 Tolerating stochastic noise

Besides the uncertainty of results among different simulation replications, stochasticity exists within one replication as well. In each trajectory produced in the simulation experiment, there may be some stochastic noises.

As shown in Figure 1, the observed model variable ‘x’ shows an oscillation property but with stochastic noises. In LTL, typically an oscillation behavior can be specified based on derivations, i.e.,

$F(((dx/dt > 0) \wedge (x > 0)) \wedge (F((dx/dt < 0) \wedge (x < 100))))$. However, with the existence of noises, the calculation of first order derivation is not applicable any longer. Alternatively, it can be expressed in LTL as

$$G(((x < 0) \rightarrow F(x > 100)) \wedge ((x > 100) \rightarrow F(x < 0)) \wedge F(x > 100)),$$

which is complex and error prone. However, it can be described with a domain specific language in a much more succinct manner by simply defining a predicate named ‘Oscillation’. What’s more, several parameters can be added to allow specifying constraints on the oscillation amplitude and period.

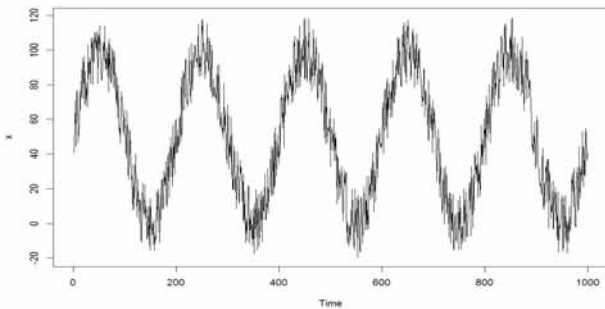


Figure 1: An example of simulation trajectory generated from experiments with a stochastic model, where the variable ‘x’ oscillates along the time with noises.

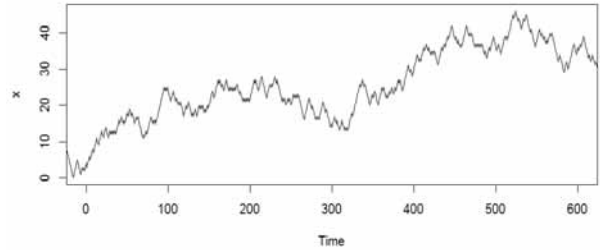


Figure 2: An example of simulation trajectory generated from experiments with a stochastic model, where the trend of x is increasing however shows some noises.

Domain specific languages can be used as an interface with easy-to-use predicates defined for the user, while those predicates can be transformed into the corresponding specification in temporal logics (depending on their semantics). In this way, the existing checking algorithm developed for temporal logics, such as [20], can be employed.

Let us look at another example. As shown in Figure 2, in this simulation trajectory generated from experiments on stochastic model, the variable ‘x’ evolves over time, exhibiting a trend of increase. However, because of the stochastic noises, it is not strictly increasing all the time, i.e., the first order derivation of ‘x’ is not always larger than zero. In this case, it is difficult to specify the increase behavior using temporal logics with noises taken into account.

On the other hand, there are different ways to tolerate the noise. In this trajectory, one can say that variable ‘x’ increases from time 0 to 200, or from time 0 to time 600, depending on how the noises are tolerated and how the increase is defined and interpreted.

Domain specific languages, which “speak” the language of the domain, provide the possibility to flexibly define specifications. Users can define the behavior property in different manners, which may not be expressible formally. In the trajectory language proposed in [16], as shown in Listing 1, several predicates are defined to describe properties, e.g., peaking, increase, decrease and reaching a steady state. Users can provide algorithms to check those predicates to tolerate the stochastic noises based on different requirements.

Taking the simulation trajectory shown in Figure 2 as example, a predicate can be that variable ‘x’ increases from time 0 to time 600.

Users can either define the semantics of increase as a simple comparison between the starting point and the ending point, where the predicate would hold. Or a more complex algorithm can be defined which checks whether the derivation of ‘x’ is within a certain a threshold, in which case the predicate may not hold because there is a relatively obvious decrease from time 200 to time 300 and the deviation could be out of the given threshold.

3.3 Robustness measurement

Besides probability estimation, robustness measurement is of interest in analysing stochastic systems. A general definition is that ‘robustness is a property that allows a system to maintain its functions against internal and external perturbations’ [21]. To formally analyze robustness, a precise definition is required and this can be performed from different perspectives.

While checking whether a behavior property holds or not provides the yes/no answer, i.e., the satisfiability, it may also be interesting and important to check to which extent the property holds, or how far it is from the property holding. Thus, the robustness degree regarding property satisfactory can be defined. There is plenty of work on this type of robustness analysis, e.g., [22], [23] and [24]. The behavior properties are first specified with temporal logics, such as LTL, Metric Temporal Logic (MTL) [25], or Signal Temporal Logic (STL) [26]. The robustness degree is measured based on definitions of distance between a simulation trajectory and the formalized properties, either in space or in time. Furthermore, for stochastic models, similar definition of robustness has been proposed in [27], where a robust satisfiability distribution for the formalized property can be estimated from that of multiple replications.

So far, this type of robustness measurement is not

supported in SESSL. However, as an internal domain specific language, it is easy to extend it. Based on existing work, a specification of robustness can be added into current SESSL by defining functions and integrating corresponding robustness measurement implementations.

Additionally, another common definition of robustness is to measure the capacity of the model maintaining the given behavior with respect to changes in model parameters [28]. SESSL, as shown in Listing 1, allows specifying model parameters in a range in addition to single values (line 5-6). This provides the possibility to define robustness regarding model parameters.

4 Conclusion

In comparison to deterministic models, experiments with stochastic models require to take stochasticity into account. This may include stochastic deviations between simulation runs as well as stochastic noises during one run. Domain-specific languages for the specification of simulation experiments such as SESSL allow handling these problems. In particular, they can employ existing powerful analysis and evaluation tools without much effort, as well as integrating new ones. As domain-specific languages provide a natural, domain-friendly way to document simulation experiments, their adoption is an important step towards the reproducibility of experiments and their results.

References

- [1] Peng D, Ewald R, Uhrmacher AM. Towards semantic model composition via experiments. In Proceedings of the 2nd ACM SIGSIM/PADS conference on Principles of advanced discrete simulation. *Conference on Principles of advanced discrete simulation*; 2014; ACM.151-162.
- [2] Maus C, Rybacki S, Uhrmacher AM. Rule-based multi-level modeling of cell biological systems. *BMC Systems Biology*. 2011; 5(1): 166. doi:10.1186/1752-0509-5-166.
- [3] Elmqvist H, Mattsson S-E. MODELICA-the next generation modeling language-an international design effort. In Proceedings of First World Congress of System Simulation. *First World Congress of System Simulation*; 1997; 1-3.
- [4] Teran-Somohano A, Dayıbaş O, Yılmaz L, et al. Toward a model-driven engineering framework for reproducible simulation experiment lifecycle management. In Proceedings of the 2014 Winter Simulation Conference. *Winter Simulation Conference*; 2014; Savannah, Georgia. 2694195: IEEE Press.2726-2737.

- [5] Leye S, Uhrmacher AM. GUISE—a tool for GUIDing simulation experiments. In Proceedings of Winter Simulation Conference. *Winter Simulation Conference*; 2012; Berlin, Germany. Winter Simulation Conference.305.
- [6] Ewald R, Uhrmacher AM. SESSL: A domain-specific language for simulation experiments. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*. 2014; 24(2): 11. doi:10.1145/2567895.
- [7] Ghosh D. *DSLs in Action*. Manning; 2011. 351.
- [8] Deursen Av, Klint P, Visser J. Domain-specific languages: an annotated bibliography. *SIGPLAN Notices*. 2000; 35(6): 26-36. doi:10.1145/352029.352035.
- [9] Miller J, Han J, Hybinette M. Using domain specific language for modeling and simulation: Scalation as a case study. In Proceedings of the 2010 Winter Simulation Conference. *Winter Simulation Conference*; 2010; IEEE.741-752.
- [10] Waltemath D, Adams R, Beard DA, et al. Minimum information about a simulation experiment (MIASE). *PLoS computational biology*. 2011; 7(4): e1001122_1001121-e1001122_1001124. doi:10.1371/journal.pcbi.1001122.
- [11] Rahmandad H, Sterman JD. Reporting guidelines for simulation-based research in social sciences. *System Dynamics Review*. 2012; 28(4): 396-411. doi:10.1002/sdr.1481.
- [12] Schutzel J, Peng D, Uhrmacher AM, et al. Perspectives on languages for specifying simulation experiments. In Proceedings of the 2014 Winter Simulation Conference. *Winter Simulation Conference*; 2014; IEEE.2836-2847. doi:10.1109/WSC.2014.7020125.
- [13] Odersky M, Spoon L, Venners B. *Programming in Scala*. Second Edition. Artima Press; 2010. 852.
- [14] Himmelspach J, Uhrmacher AM. Plug'n simulate. In Proceedings of the 40th Annual Simulation Symposium. *Simulation Symposium*; 2007; Norfolk, Virginia. IEEE.137-143. doi:10.1109/ANSS.2007.34
- [15] Clarke EM, Grumberg O, Peled D. *Model checking*. 1999.
- [16] Warnke T, Peng D, Haack F, et al. Towards a Language for Specifying Properties of Simulation Trajectories. In Proceedings of 12th International Conference on Computational Methods in Systems Biology. *Computational Methods in Systems Biology*; 2014; London.
- [17] Younes HL, Kwiatkowska M, Norman G, et al. Numerical vs. statistical probabilistic model checking. *International Journal on Software Tools for Technology Transfer*. 2006; 8(3): 216-228. doi:10.1007/978-3-540-24730-2_4.
- [18] Jha SK, Clarke EM, Langmead CJ, et al. A bayesian approach to model checking biological systems. In *Computational Methods in Systems Biology*; 2009; Springer.218-234.
- [19] Wald A. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*. 1945; 16(2): 117-186. doi:doi:10.1214/aoms/1177731118.
- [20] Spieler D. Model checking of oscillatory and noisy periodic behavior in Markovian population models [Doctoral dissertation]. Saarland University; 2009.
- [21] Kitano H. Towards a theory of biological robustness. *Molecular systems biology*. 2007; 3(1): 137. doi:10.1038/msb4100179.
- [22] Rizk A, Batt G, Fages F, et al. On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In *Computational Methods in Systems Biology*; 2008; Springer.251-268.
- [23] Donzé A, Maler O. Robust satisfaction of temporal logic over real-valued signals. Springer; 2010.
- [24] Fainekos GE, Pappas GJ (2006). Robustness of temporal logic specifications for finite state sequences in metric spaces, Technical Report MS-CIS-06-05, Dept. of CIS, Univ. of Pennsylvania.
- [25] Koymans R. Specifying real-time properties with metric temporal logic. *Real-time systems*. 1990; 2(4): 255-299. doi:10.1007/BF01995674.
- [26] Maler O, Nickovic D. Monitoring temporal properties of continuous signals. *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer; 2004. 152-166.
- [27] Bartocci E, Bortolussi L, Nenzi L, et al. On the robustness of temporal properties for stochastic models. *Electronic Proceedings in Theoretical Computer Science*. 2013; 3-19. doi:10.4204/EPTCS.125.1.
- [28] Komorowski M, Costa MJ, Rand DA, et al. Sensitivity, robustness, and identifiability in stochastic chemical kinetics models. *Proceedings of the National Academy of Sciences*. 2011; 108(21): 8645-8650. doi:10.1073/pnas.1015814108

SNE Simulation News

EUROSIM Data and Quick Info



EUROSIM 2016

9th EUROSIM Congress on Modelling and Simulation

City of Oulu, Finland, September 16-20, 2016

www.eurosim.info

Contents

Info EUROSIM	2
Info EUROSIM Societies	3 - 8
Info ASIM, CAE-SMSG	3
Info CROSSIM, CSSS, DBSS, FRANCOSIM	4
Info HSS, ISCS, LIOPHANT.....	5
Info LSS, PSCS, SIMS, SLOSIM.....	6
Info UKSIM, KA-SIM, ROMSIM	7
Info RNSS, Info SNE	8

Simulation Notes Europe SNE is the official membership journal of EUROSIM and distributed / available to members of the EUROSIM Societies as part of the membership benefits. SNE is published in a printed version (Print ISSN 2305-9974) and in an online version (Online ISSN 2306-0271). With Online SNE the publisher ARGESIM follows the Open Access strategy for basic SNE contributions. Since 2012 Online SNE contributions are identified by DOI 10.11128/sne.xx.nnnnn. for better web availability and indexing.

Print SNE, high-resolution Online SNE, and additional SNE contributions are available via membership in a EUROSIM society.

This *EUROSIM Data & Quick Info* compiles data from EUROSIM societies and groups: addresses, weblinks, officers of societies with function and email, to be published regularly in SNE issues.

SNE Reports Editorial Board

EUROSIM Esko Juuso, esko.juuso@oulu.fi

Borut Zupančič, borut.zupancic@fe.uni-lj.si

Felix Breitenecker, Felix.Breitenecker@tuwien.ac.at

ASIM Thorsten Pawletta, pawel@mb.hs-wismar.de

CAE-SMSG Emilio Jiminez, emilio.jiminez@unirioja.es

CROSSIM Vesna Dušak, vdusak@foi.hr

CSSS Mikuláš Alexík, alexik@frtk.utc.sk

DBSS A. Heemink, a.w.heemink@its.tudelft.nl

FRANCOSIM Karim Djouani, djouani@u-pec.fr

HSS András Jávör, javor@eik.bme.hu

ISCS M. Savastano, mario.savastano@unina.it

LIOPHANT F. Longo, f.longo@unicat.it

LSS Yuri Merkurjev, merkur@itl.rtu.lv

PSCS Zenon Sosnowski, zenon@ii.pb.bialystok.pl

SIMS Esko Juuso, esko.juuso@oulu.fi

SLOSIM Rihard Karba, rihard.karba@fe.uni-lj.si

UKSIM Richard Zobel, r.zobel@ntlworld.com

KA-SIM Edmond Hajrizi, info@ka-sim.com

ROMSIM Florin Stanculescu, sflorin@ici.ro

RNSS Y. Senichenkov, sneyb@dcn.infos.ru

SNE Editorial Office /ARGESIM

→ www.sne-journal.org, www.eurosim.info

✉ office@sne-journal.org (info, news)

✉ eic@sne-journal.org Felix Breitenecker (publications)

If you have any information, announcement, etc. you want to see published, please contact a member of the editorial board in your country or the editorial office. For scientific publications, please contact the EiC.



EUROSIM Federation of European Simulation Societies

General Information. EUROSIM, the Federation of European Simulation Societies, was set up in 1989. The purpose of EUROSIM is to provide a European forum for simulation societies and groups to promote advancement of modelling and simulation in industry, research, and development. → www.eurosim.info

Member Societies. EUROSIM members may be national simulation societies and regional or international societies and groups dealing with modelling and simulation. At present EUROSIM has fourteen *Full Members* and three *Observer Members*:

ASIM	Arbeitsgemeinschaft Simulation <i>Austria, Germany, Switzerland</i>
CEA-SMSG	Spanish Modelling and Simulation Group <i>Spain</i>
CROSSIM	Croatian Society for Simulation Modeling <i>Croatia</i>
CSSS	Czech and Slovak Simulation Society <i>Czech Republic, Slovak Republic</i>
DBSS	Dutch Benelux Simulation Society <i>Belgium, Netherlands</i>
FRANCO-SIM	Société Francophone de Simulation <i>Belgium, France</i>
HSS	Hungarian Simulation Society <i>Hungary</i>
ISCS	Italian Society for Computer Simulation <i>Italy</i>
LIOPHANT	LIOPHANT Simulation Club <i>Italy & International, Observer Member</i>
LSS	Latvian Simulation Society <i>Latvia</i>
PSCS	Polish Society for Computer Simulation <i>Poland</i>
SIMS	Simulation Society of Scandinavia <i>Denmark, Finland, Norway, Sweden</i>
SLOSIM	Slovenian Simulation Society <i>Slovenia</i>
UKSIM	United Kingdom Simulation Society <i>UK, Ireland</i>
KA-SIM	Romanian Society for Modelling and Simulation, <i>Romania, Observer Member</i>
ROMSIM	Romanian Society for Modelling and Simulation, <i>Romania, Observer Member</i>
RNSS	Russian National Simulation Society <i>Russian Federation, Observer Member</i>

EUROSIM Board / Officers. EUROSIM is governed by a board consisting of one representative of each member society, president and past president, and representatives for SNE Simulation notes Europe. The President is nominated by the society organising the next EUROSIM Congress. Secretary and Treasurer are elected out of members of the Board.

President	Esko Juuso (SIMS) <i>esko.juuso@oulu.fi</i>
Past President	Khalid Al.Begain (UKSIM) <i>kbegain@glam.ac.uk</i>
Secretary	Borut Zupančič (SLOSIM) <i>borut.zupancic@fe.uni-lj.si</i>
Treasurer	Felix Breitenecker (ASIM) <i>felix.breitenecker@tuwien.ac.at</i>
SNE Repres.	Felix Breitenecker <i>felix.breitenecker@tuwien.ac.at</i>

SNE – Simulation Notes Europe. SNE is a scientific journal with reviewed contributions as well as a membership newsletter for EUROSIM with information from the societies in the *News Section*. EUROSIM societies are offered to distribute to their members the journal SNE as official membership journal. SNE Publishers are EUROSIM, ARGESIM and ASIM.

Editor-in-chief	Felix Breitenecker <i>felix.breitenecker@tuwien.ac.at</i>
-----------------	--

→ www.sne-journal.org,

✉ office@sne-journal.org

EUROSIM Congress. EUROSIM is running the triennial conference series EUROSIM Congress. The congress is organised by one of the EUROSIM societies.

EUROSIM 2016 will be organised by SIMS in Oulu, Finland, September 16-20, 2016.

Chairs / Team EUROSIM 2016

Esko Juuso EUROSIM President, esko.juuso@oulu.fi
Erik Dahlquist SIMS President, erik.dahlquist@mdh.se
Kauko Leiviskä EUROSIM 2016 Chair,
kauko.leiviska@oulu.fi

→ www.eurosim.info

✉ office@automaatioseura.fi



EUROSIM Member Societies



ASIM German Simulation Society Arbeitsgemeinschaft Simulation

ASIM (Arbeitsgemeinschaft Simulation) is the association for simulation in the German speaking area, servicing mainly Germany, Switzerland and Austria. ASIM was founded in 1981 and has now about 700 individual members, and 30 institutional or industrial members.

→ www.asim-gi.org with members' area

✉ info@asim-gi.org, admin@asim-gi.org

✉ ASIM – Inst. f. Analysis and Scientific Computing
Vienna University of Technology
Wiedner Hauptstraße 8-10, 1040 Vienna, Austria

ASIM Officers

President	Felix Breitenecker felix.breitenecker@tuwien.ac.at
Vice presidents	Sigrid Wenzel, s.wenzel@uni-kassel.de T. Pawletta, pawel@mb.hs-wismar.de
Secretary	Ch. Deatcu, christina.deatcu@hs-wismar.de
Treasurer	Anna Mathe, anna.mathe@tuwien.ac.at
Membership Affairs	S. Wenzel, s.wenzel@uni-kassel.de W. Maurer, werner.maurer@zhwin.ch Ch. Deatcu, christina.deatcu@hs-wismar.de F. Breitenecker, felix.breitenecker@tuwien.ac.at
Universities / Research Inst.	S. Wenzel, s.wenzel@uni-kassel.de W. Wiechert, WWiechert@fz-juelich.de J. Haase, Joachim.Haase@eas.iis.fraunhofer.de Katharina Nöh, k.noeh@fz-juelich.de
Industry	S. Wenzel, s.wenzel@uni-kassel.de K. Panreck, Klaus.Panreck@hella.com
Conferences	Klaus Panreck Klaus.Panreck@hella.com J. Wittmann, wittmann@htw-berlin.de
Publications	Th. Pawletta, pawel@mb.hs-wismar.de Christina Deatcu, christina.deatcu@hs-wismar.de F. Breitenecker, felix.breitenecker@tuwien.ac.at
Repr. EUROSIM	F. Breitenecker, felix.breitenecker@tuwien.ac.at N. Popper, niki.popper@drahtwarenhandlung.at
Education / Teaching	A. Körner, andreas.koerner@tuwien.ac.at N. Popper, niki.popper@drahtwarenhandlung.at Katharina Nöh, k.noeh@fz-juelich.de
International Affairs	A. Körner, andreas.koerner@tuwien.ac.at O. Rose, Oliver.Rose@tu-dresden.de
Editorial Board SNE	T. Pawletta, pawel@mb.hs-wismar.de Ch. Deatcu, christina.deatcu@hs-wismar.de
Web EUROSIM	Anna Mathe, anna.mathe@tuwien.ac.at

Last data update December 2013

ASIM Working Committee. ASIM, part of GI - Gesellschaft für Informatik, is organised in Working Committees, dealing with applications and comprehensive subjects in modelling and simulation:

ASIM Working Committee

GMMS	Methods in Modelling and Simulation Th. Pawletta, pawel@mb.hs-wismar.de
SUG	Simulation in Environmental Systems Wittmann, wittmann@informatik.uni-hamburg.de
STS	Simulation of Technical Systems H.T.Mammen, Heinz-Theo.Mammen@hella.com
SPL	Simulation in Production and Logistics Sigrid Wenzel, s.wenzel@uni-kassel.de
Edu	Simulation in Education/Education in Simulation N. Popper, niki.popper@dwh.at A. Körner, andreas.koerner@tuwien.ac.at
	Working Groups for Simulation in Business Administration, in Traffic Systems, for Standardisation, for Validation, etc.

CEA-SMSG – Spanish Modelling and Simulation Group

CEA is the Spanish Society on Automation and Control. In order to improve the efficiency and to deep into the different fields of automation, the association is divided into thematic groups, one of them is named 'Modelling and Simulation', constituting the group.

→ www.cea-ifac.es/wwwgrupos/simulacion

→ simulacion@cea-ifac.es

✉ CEA-SMSG / María Jesús de la Fuente,
System Engineering and Automatic Control department,
University of Valladolid,
Real de Burgos s/n., 47011 Valladolid, SPAIN

CAE - SMSG Officers

President	M. A. Piera Eroles, MiquelAngel.Piera@uab.es
Vice president	Emilio Jimenez, emilio.jimenez@unirioja.es
Repr. EUROSIM	Emilio Jimenez, emilio.jimenez@unirioja.es
Edit. Board SNE	Emilio Jimenez, emilio.jimenez@unirioja.es
Web EUROSIM	Mercedes Peres, mercedes.perez@unirioja.es

Last data update December 2013



CROSSIM – Croatian Society for Simulation Modelling

CROSSIM-Croatian Society for Simulation Modelling was founded in 1992 as a non-profit society with the goal to promote knowledge and use of simulation methods and techniques and development of education. CROSSIM is a full member of EUROSIM since 1997.

→ www.eurosim.info

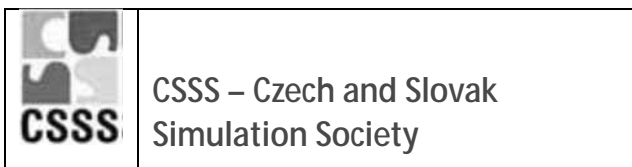
✉ vdusak@foi.hr

✉ CROSSIM / Vesna Dušak
Faculty of Organization and
Informatics Varaždin, University of Zagreb
Pavlinska 2, HR-42000 Varaždin, Croatia

CROSSIM Officers

President	Vesna Dušak, vdusak@foi.hr
Vice president	Jadranka Božikov, jbozikov@snz.hr
Secretary	Vesna Bosilj-Vukšić, vbosilj@efzg.hr
Executive board members	Vlatko Čerić, vceric@efzg.hr Tarzan Legović, legovic@irb.hr
Repr. EUROSIM	Jadranka Božikov, jbozikov@snz.hr
Edit. Board SNE	Vesna Dušak, vdusak@foi.hr
Web EUROSIM	Jadranka Božikov, jbozikov@snz.hr

Last data update December 2012



CSSS -The Czech and Slovak Simulation Society has about 150 members working in Czech and Slovak national scientific and technical societies (*Czech Society for Applied Cybernetics and Informatics, Slovak Society for Applied Cybernetics and Informatics*). The main objectives of the society are: development of education and training in the field of modelling and simulation, organising professional workshops and conferences, disseminating information about modelling and simulation activities in Europe. Since 1992, CSSS is full member of EUROSIM.

→ www.fit.vutbr.cz/CSSS

✉ snorek@fel.cvut.cz

✉ CSSS / Miroslav Šnorek, CTU Prague
FEE, Dept. Computer Science and Engineering,
Karlovo nám. 13, 121 35 Praha 2, Czech Republic

CSSS Officers

President	Miroslav Šnorek, snorek@fel.cvut.cz
Vice president	Mikuláš Alexík, alexik@frtk.fri.utc.sk
Treasurer	Evžen Kindler, ekindler@centrum.cz
Scientific Secr.	A. Kavička, Antonin.Kavicka@upce.cz
Repr. EUROSIM	Miroslav Šnorek, snorek@fel.cvut.cz
Deputy	Mikuláš Alexík, alexik@frtk.fri.utc.sk
Edit. Board SNE	Mikuláš Alexík, alexik@frtk.fri.utc.sk
Web EUROSIM	Petr Peringer, peringer@fit.vutbr.cz

Last data update December 2012

DBSS – Dutch Benelux Simulation Society

The Dutch Benelux Simulation Society (DBSS) was founded in July 1986 in order to create an organisation of simulation professionals within the Dutch language area. DBSS has actively promoted creation of similar organisations in other language areas. DBSS is a member of EUROSIM and works in close cooperation with its members and with affiliated societies.

→ www.eurosim.info

✉ a.w.heemink@its.tudelft.nl

✉ DBSS / A. W. Heemink
Delft University of Technology, ITS - twi,
Mekelweg 4, 2628 CD Delft, The Netherlands

DBSS Officers

President	A. Heemink, a.w.heemink@its.tudelft.nl
Vice president	W. Smit, smitnet@wxs.nl
Treasurer	W. Smit, smitnet@wxs.nl
Secretary	W. Smit, smitnet@wxs.nl
Repr. EUROSIM	A. Heemink, a.w.heemink@its.tudelft.nl
Deputy	W. Smit, smitnet@wxs.nl
Edit. Board SNE	A. Heemink, a.w.heemink@its.tudelft.nl

Last data update April 2006

FRANCOSIM – Société Francophone de Simulation

FRANCOSIM was founded in 1991 and aims to the promotion of simulation and research, in industry and academic fields. Francosim operates two poles.

- Pole Modelling and simulation of discrete event systems. Pole Contact: *Henri Pierreval, pierreval@imfa.fr*
- Pole Modelling and simulation of continuous systems. Pole Contact: *Yskandar Hamam, y.hamam@esiee.fr*

→ www.eurosim.info✉ y.hamam@esiee.fr

✉ FRANCOSIM / Yskandar Hamam
Groupe ESIEE, Cité Descartes,
BP 99, 2 Bd. Blaise Pascal,
93162 Noisy le Grand CEDEX, France

FRANCOSIM Officers

President	Karim Djouani, djouani@u-pec.fr
Treasurer	François Rocaries, f.rocaries@esiee.fr
Repr. EUROSIM	Karim Djouani, djouani@u-pec.fr
Edit. Board SNE	Karim Djouani, djouani@u-pec.fr

*Last data update December 2012***HSS – Hungarian Simulation Society**

The Hungarian Member Society of EUROSIM was established in 1981 as an association promoting the exchange of information within the community of people involved in research, development, application and education of simulation in Hungary and also contributing to the enhancement of exchanging information between the Hungarian simulation community and the simulation communities abroad. HSS deals with the organization of lectures, exhibitions, demonstrations, and conferences.

→ www.eurosim.info✉ javor@eik.bme.hu

✉ HSS / András Jávör,
Budapest Univ. of Technology and Economics,
Sztoczek u. 4, 1111 Budapest, Hungary

HSS Officers

President	András Jávör, javor@eik.bme.hu
Vice president	Gábor Szűcs, szucs@itm.bme.hu
Secretary	Ágnes Vigh, vigh@itm.bme.hu
Repr. EUROSIM	András Jávör, javor@eik.bme.hu
Deputy	Gábor Szűcs, szucs@itm.bme.hu
Edit. Board SNE	András Jávör, javor@eik.bme.hu
Web EUROSIM	Gábor Szűcs, szucs@itm.bme.hu

*Last data update March 2008***ISCS – Italian Society for Computer Simulation**

The Italian Society for Computer Simulation (ISCS) is a scientific non-profit association of members from industry, university, education and several public and research institutions with common interest in all fields of computer simulation.

→ www.eurosim.info✉ Mario.savastano@uniina.at

✉ ISCS / Mario Savastano,
c/o CNR - IRSIP,
Via Claudio 21, 80125 Napoli, Italy

ISCS Officers

President	M. Savastano, mario.savastano@uniina.it
Vice president	F. Maceri, Franco.Maceri@uniroma2.it
Repr. EUROSIM	F. Maceri, Franco.Maceri@uniroma2.it
Secretary	Paola Provenzano, paola.provenzano@uniroma2.it
Edit. Board SNE	M. Savastano, mario.savastano@uniina.it

Last data update December 2010**LIOPHANT Simulation**

Liophant Simulation is a non-profit association born in order to be a trait-d'union among simulation developers and users; Liophant is devoted to promote and diffuse the simulation techniques and methodologies; the Association promotes exchange of students, sabbatical years, organization of International Conferences, organization of courses and stages in companies to apply the simulation to real problems.

→ www.liophant.org✉ info@liophant.org

✉ LIOPHANT Simulation, c/o Agostino G. Bruzzone,
DIME, University of Genoa, Polo Savonese,
via Molinero 1, 17100 Savona (SV), Italy

LIOPHANT Officers

President	A.G. Bruzzone, agostino@itim.unige.it
Director	E. Bocca, enrico.bocca@liophant.org
Secretary	A. Devoti, devoti.a@iveco.com
Treasurer	Marina Masseimassei@itim.unige.it
Repr. EUROSIM	A.G. Bruzzone, agostino@itim.unige.it
Deputy	F. Longo, f.longo@unical.it
Edit. Board SNE	F. Longo, f.longo@unical.it
Web EUROSIM	F. Longo, f.longo@unical.it

Last data update December 2013



LSS – Latvian Simulation Society

The Latvian Simulation Society (LSS) has been founded in 1990 as the first professional simulation organisation in the field of Modelling and simulation in the post-Soviet area. Its members represent the main simulation centres in Latvia, including both academic and industrial sectors.

→ briedis.itl.rtu.lv/imb/

✉ merkur@itl.rtu.lv

✉ LSS / Yuri Merkuryev, Dept. of Modelling and Simulation Riga Technical University
Kalku street 1, Riga, LV-1658, LATVIA

LSS Officers

President	Yuri Merkuryev, merkur@itl.rtu.lv
Secretary	Artis Teilans, Artis.Teilans@exigenservices.com
Repr. EUROSIM	Yuri Merkuryev, merkur@itl.rtu.lv
Deputy	Artis Teilans, Artis.Teilans@exigenservices.com
Edit. Board SNE	Yuri Merkuryev, merkur@itl.rtu.lv
Web EUROSIM	Oksana Sosho, oksana@itl.rtu.lv

Last data update December 2013

PSCS – Polish Society for Computer Simulation

PSCS was founded in 1993 in Warsaw. PSCS is a scientific, non-profit association of members from universities, research institutes and industry in Poland with common interests in variety of methods of computer simulations and its applications. At present PSCS counts 257 members.

→ www.ptsk.man.bialystok.pl

✉ leon@ibib.waw.pl

✉ PSCS / Leon Bobrowski, c/o IBIB PAN,
ul. Trojdena 4 (p.416), 02-109 Warszawa, Poland

PSCS Officers

President	Leon Bobrowski, leon@ibib.waw.pl
Vice president	Tadeusz Nowicki, Tadeusz.Nowicki@wat.edu.pl
Treasurer	Z. Sosnowski, zenon@ii.pb.bialystok.pl
Secretary	Zdzislaw Galkowski, Zdzislaw.Galkowski@simr.pw.edu.pl
Repr. EUROSIM	Leon Bobrowski, leon@ibib.waw.pl
Deputy	Tadeusz Nowicki, tadeusz.nowicki@wat.edu.pl
Edit. Board SNE	Zenon Sosnowski, z.sosnowski@pb.edu.pl
Web EUROSIM	Magdalena Topczewska m.topczewska@pb.edu.pl

Last data update December 2013

SIMS – Scandinavian Simulation Society

SIMS is the *Scandinavian Simulation Society* with members from the four Nordic countries Denmark, Finland, Norway and Sweden. The SIMS history goes back to 1959. SIMS practical matters are taken care of by the SIMS board consisting of two representatives from each Nordic country (Iceland one board member).

SIMS Structure. SIMS is organised as federation of regional societies. There are FinSim (Finnish Simulation Forum), DKSIM (Dansk Simuleringsforening) and NFA (Norsk Forening for Automatisering).

→ www.scansims.org

✉ esko.juuso@oulu.fi

✉ SIMS / Esko Juuso, Department of Process and Environmental Engineering, 90014 Univ.Oulu, Finland

SIMS Officers

President	Esko Juuso, esko.juuso@oulu.fi
Vice president	Erik Dahlquist, erik.dahlquist@mdh.se
Treasurer	Vadim Engelson, vadim.engelson@mathcore.com
Repr. EUROSIM	Esko Juuso, esko.juuso@oulu.fi
Edit. Board SNE	Esko Juuso, esko.juuso@oulu.fi
Web EUROSIM	Vadim Engelson, vadim.engelson@mathcore.com

Last data update December 2013



SLOSIM – Slovenian Society for Simulation and Modelling

SLOSIM - Slovenian Society for Simulation and Modelling was established in 1994 and became the full member of EUROSIM in 1996. Currently it has 69 members from both slovenian universities, institutes, and industry. It promotes modelling and simulation approaches to problem solving in industrial as well as in academic environments by establishing communication and cooperation among corresponding teams.

→ www.slosim.si

✉ slosim@fe.uni-lj.si

✉ SLOSIM / Rihard Karba, Faculty of Electrical Engineering, University of Ljubljana,
Tržaška 25, 1000 Ljubljana, Slovenia

**SLOSIM Officers**

President	Vito Logar, vito.logar@fe.uni-lj.si
Vice president	Božidar Šarler, bozidar.sarler@ung.si
Secretary	Aleš Belič, ales.belic@sandoz.com
Treasurer	Milan Simčič, milan.simcic@fe.uni-lj.si
Repr. EUROSIM	B. Zupančič, borut.zupancic@fe.uni-lj.si
Deputy	Vito Logar, vito.logar@fe.uni-lj.si
Edit. Board SNE	Rihard Karba, rihard.karba@fe.uni-lj.si
Web EUROSIM	Vito Logar, vito.logar@fe.uni-lj.si

*Last data update December 2013***UKSIM - United Kingdom Simulation Society**

UKSIM has more than 100 members throughout the UK from universities and industry. It is active in all areas of simulation and it holds a biennial conference as well as regular meetings and workshops.

→ www.uksim.org.uk✉ david.al-dabass@ntu.ac.uk

✉ UKSIM / Prof. David Al-Dabass
Computing & Informatics,
Nottingham Trent University
Clifton lane, Nottingham, NG11 8NS
United Kingdom

UKSIM Officers

President	David Al-Dabass, david.al-dabass@ntu.ac.uk
Vice president	A. Orsoni, A.Orsoni@kingston.ac.uk
Secretary	Richard Cant, richard.cant@ntu.ac.uk
Treasurer	A. Orsoni, A.Orsoni@kingston.ac.uk
Membership chair	K. Al-Begain, kbegain@glam.ac.uk
Univ. liaison chair	R. Cheng, rhc@maths.soton.ac.uk
Repr. EUROSIM	Richard Zobel, r.zobel@ntlworld.com
Deputy	K. Al-Begain, kbegain@glam.ac.uk
Edit. Board SNE	Richard Zobel, r.zobel@ntlworld.com

*Last data update December 2013***EUROSIM OBSERVER MEMBERS****KA-SIM Kosovo Simulation Society**

Kosova Association for Modeling and Simulation (KA – SIM, founded in 2009), is part of Kosova Association of Control, Automation and Systems Engineering (KA – CASE). KA – CASE was registered in 2006 as non Profit Organization and since 2009 is National Member of IFAC – International Federation of Automatic Control. KA-SIM joined EUROSIM as Observer Member in 2011.

KA-SIM has about 50 members, and is organizing the international conference series International Conference in Business, Technology and Innovation, in November, in Durrhës, Albania, an IFAC Simulation workshops in Pristina.

→ www.ubt-uni.net/ka-case✉ ehajrizi@ubt-uni.net

✉ MOD&SIM KA-CASE
Att. Dr. Edmond Hajrizi
Univ. for Business and Technology (UBT)
Lagjja Kalabria p.n., 10000 Prishtina, Kosovo

KA-SIM Officers

President	Edmond Hajrizi, ehajrizi@ubt-uni.net
Vice president	Muzafer Shala, info@ka-sim.com
Secretary	Lulzim Beqiri, info@ka-sim.com
Treasurer	Selman Berisha, info@ka-sim.com
Repr. EUROSIM	Edmond Hajrizi, ehajrizi@ubt-uni.net
Deputy	Muzafer Shala, info@ka-sim.com
Edit. Board SNE	Edmond Hajrizi, ehajrizi@ubt-uni.net
Web EUROSIM	Betim Gashi, info@ka-sim.com

*Last data update December 2013***ROMSIM – Romanian Modelling and Simulation Society**

ROMSIM has been founded in 1990 as a non-profit society, devoted to theoretical and applied aspects of modelling and simulation of systems. ROMSIM currently has about 100 members from Romania and Moldavia.

→ www.ici.ro/romsim/✉ sflorin@ici.ro

✉ ROMSIM / Florin Stanculescu,
National Institute for Research in Informatics, Averescu
Av. 8 – 10, 71316 Bucharest, Romania



ROMSIM Officers	
President	Florin Stanciulescu, sflorin@ici.ro
Vice president	Florin Hartescu, flory@ici.ro Marius Radulescu, mradulescu@ici.ro
Repr. EUROSIM	Florin Stanciulescu, sflorin@ici.ro
Deputy	Marius Radulescu, mradulescu@ici.ro
Edit. Board SNE	Florin Stanciulescu, sflorin@ici.ro
Web EUROSIM	Zoe Radulescu, radulescu@ici.ro

Last data update December 2012

RNSS – Russian Simulation Society

NSS - The Russian National Simulation Society (Национальное Общество Имитационного Моделирования – НОИМ) was officially registered in Russian Federation on February 11, 2011. In February 2012 NSS has been accepted as an observer member of EUROSIM.

→ www.simulation.su

✉ yusupov@iias.spb.su

✉ RNSS / R. M. Yusupov,
St. Petersburg Institute of Informatics and Automation
RAS, 199178, St. Petersburg, 14th lin. V.O, 39

RNSS Officers	
President	R. M. Yusupov, yusupov@iias.spb.su
Chair Man. Board	A. Plotnikov, plotnikov@sstc.spb.ru
Secretary	M. Dolmatov, dolmatov@simulation.su
Repr. EUROSIM	R. M. Yusupov, yusupov@iias.spb.su
Deputy	B. Sokolov, sokol@iias.spb.su
Edit. Board SNE	Y. Senichenkov, sneyb@dcn.infos.ru

Last data update February 2012

SNE – Simulation Notes Europe

Simulation Notes Europe publishes peer reviewed *Technical Notes*, *Short Notes* and *Overview Notes* on developments and trends in modelling and simulation in various areas and in application and theory. Furthermore SNE documents the ARGESIM Benchmarks on *Modeling Approaches and Simulation Implementations* with publication of definitions, solutions and discussions (*Benchmark Notes*). Special *Educational Notes* present the use of modelling and simulation in and for education and for e-learning.

SNE is the official membership journal of EUROSIM, the Federation of European Simulation Societies. A News Section in SNE provides information for EUROSIM Simulation Societies and Simulation Groups. In 2013, SNE introduced an extended submission strategy i) individual submissions of scientific papers, and ii) submissions of selected contributions from conferences of EUROSIM societies for post-conference publication (suggested by conference organizer and authors) – both with peer review.

SNE is published in a printed version (Print ISSN 2305-9974) and in an online version (Online ISSN 2306-0271). With Online SNE the publisher ARGESIM follows the Open Access strategy, allowing download of published contributions for free. Since 2012 Online SNE contributions are identified by an DOI (Digital Object Identifier) assigned to the publisher ARGESIM (DOI prefix 10.11128). Print SNE, high-resolution Online SNE, source codes of the *Benchmarks* and other additional sources are available for subscription via membership in a EUROSIM society.

Authors Information. Authors are invited to submit contributions which have not been published and have not been considered for publication elsewhere to the SNE Editorial Office. SNE distinguishes different types of contributions (*Notes*):

- *Overview Note* – State-of-the-Art report in a specific area, up to 14 pages, only upon invitation
- *Technical Note* – scientific publication on specific topic in modelling and simulation, 6 – 8 (10) pages
- *Education Note* – modelling and simulation in / for education and e-learning; max. 6 pages
- *Short Note* – recent development on specific topic, max. 4 pages
- *Software Note* – specific implementation with scientific analysis, max 4 pages
- *Benchmark Note* – Solution to an ARGESIM Benchmark; basic solution 2 pages, extended and commented solution 4 pages, comparative solutions on invitation

Interested authors may find further information at SNE's website → www.sne-journal.org (layout templates for *Notes*, requirements for benchmark solutions, etc.).

SNE Editorial Office /ARGESIM

→ www.sne-journal.org, www.eurosim.info

✉ office@sne-journal.org (info, news)

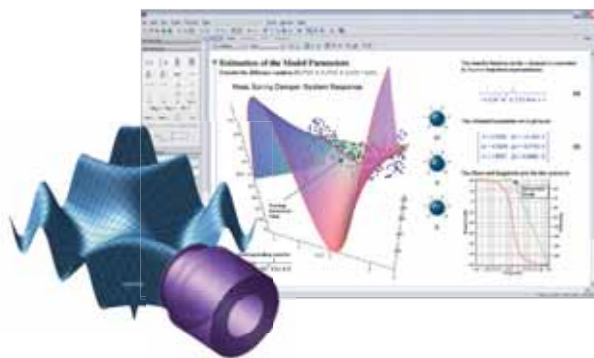
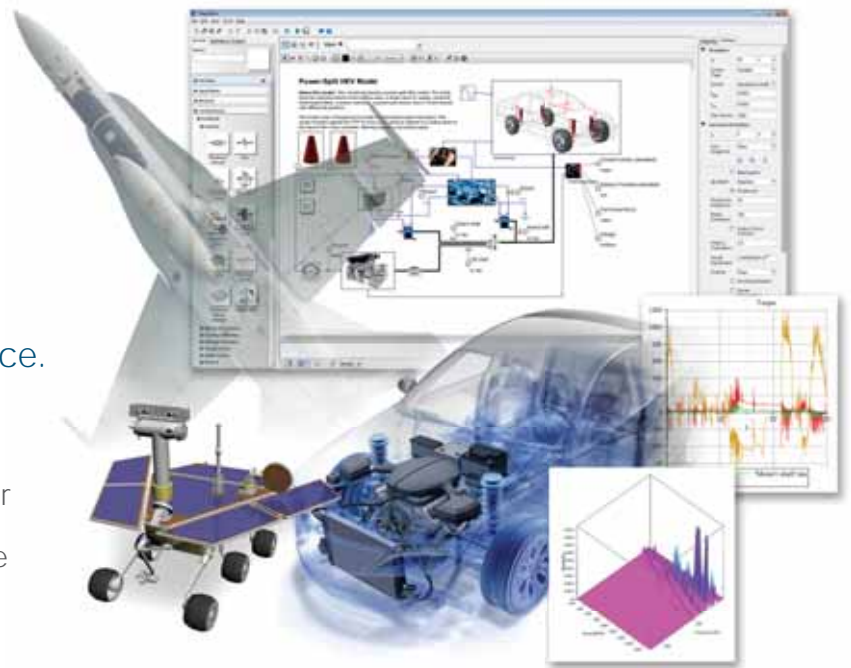
✉ eic@sne-journal.org Felix Breitenecker
(publications)

A modern approach to modeling and simulation



With MapleSim, educators have an industry-proven tool to help bridge the gap between theory and practice.

- MapleSim illustrates concepts, and helps students learn the connection between theory and physical behavior
- A wide variety of models are available to help get started right away



Maple™

MapleSim is built on Maple, which combines the world's most powerful mathematical computation engine with an intuitive, "clickable" user interface.

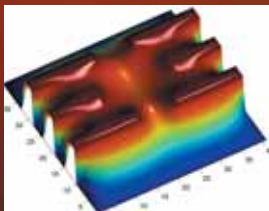
To learn more about how you can reinforce engineering concepts using a combination of theory, simulation, and hardware, view this webinar.

www.maplesoft.com/SNEWebinar

Contact us: +49 (0)241/980919-30

Parlez-vous MATLAB?

Über eine Million Menschen weltweit sprechen MATLAB. Ingenieure und Wissenschaftler in allen Bereichen – von der Luft- und Raumfahrt über die Halbleiterindustrie bis zur Biotechnologie, Finanzdienstleistungen und Geo- und Meereswissenschaften – nutzen MATLAB, um ihre Ideen auszudrücken. Sprechen Sie MATLAB?



Modellierung eines elektrischen Potentials in einem Quantum Dot.

*Dieses Beispiel finden Sie unter:
www.mathworks.de/tc*

MATLAB[®]
The language of technical computing

