

SNE SIMULATION NOTES EUROPE



Journal on Developments and Trends in Modelling and Simulation

EUROSIM Scientific Membership Journal

Vol. 33 No.4, December 2023

ISSN Online 2306-0271

DOI 10.11128/sne.33.4.1066

ISSN Print 2305-9974

ARGESIM



**ASIM Workshop Simulation in den Umwelt-und Geowissenschaften
Leipzig, April 10-12, 2024**

Schwerpunktthema des Workshops 2024 lautet
"Programmieren als Werkzeug für Wenig-Programmierer"

fa-ui.gi.de/veranstaltung/workshop-simulation-in-den-umwelt-und-geowissenschaften-1



**ASIM SST 2024, 27th Symposium Simulation Technique
Munich, September 4-6, 2024**

2024 Winter Simulation Conference, December 15-18, 2024, Orlando, Florida



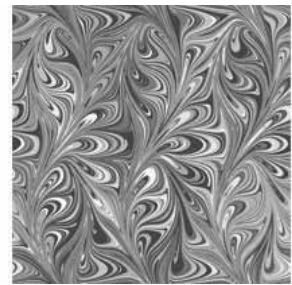
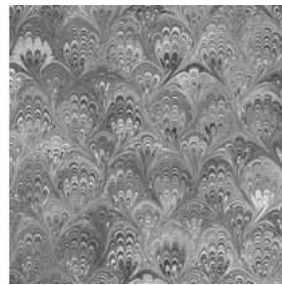
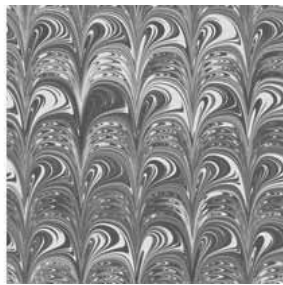
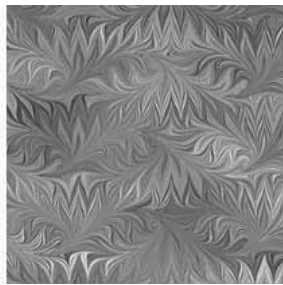
EUROSIM Congress 2026
July 2026, Italy www.eurosim.info

Editorial

Dear Readers, This last issue of 2023, SNE 33(4), publishes seven contributions of mainly software related topics. We are glad, to continue with three ‘Software Notes’, presenting the impact of simulation software development for modelling. D. Jammer, P. Junglas, T. Pawletta, and S. Pawletta present a simulator and applications for NSA-DEVS; T. Pawletta and J. Bartelt outline RL and DES with a new experimental frame concept within a case study in MATLAB/SimEvents; and P. Junglas reports on the implementation of thermodynamic cyclic processes by the DLR ThermoFluid Stream Library. Three Technical Notes deal with a statistical model between cardiac parameters and driver fatigue (M. Bachler et al.), with testing for AI-based systems in aviation (B. Lukić et al.), and with use cases deploying DES and VR for digital tools for craft professionals (B. Prell, J. Reiff-Stephan) – all with software emphasis. For SNE Volume 33, Graham Horton from University Magdeburg, provided his computer-generated marbled pattern graphics. Graham Horton started modelling and simulating the handcraft of making marbled paper, used for covers in traditional bookbinding. The algorithms in behind manufacture not only approximations of handmade marbled patterns, they generate new marbled patterns as type of algorithmic art: Digital Marbling. Below the cover pictures for SNE issues 2023, of type ‘Bird Wing’, ‘Italian Sisters’, ‘Bouquet’, and ‘Serpentine’ (this issue) – many thanks to him.

I would like to thank all authors for their contributions for this issue, also and many thanks to the SNE Editorial Office for layout, typesetting, preparations for printing, electronic publishing, and much more.

Felix Breitenecker, SNE Editor-in-Chief, eic@sne-journal.org; felix.breitenecker@tuwien.ac.at



Contents SNE 33(4)

e-SNE 33(4), DOI 10.11128/sne.33.4.1066

www.sne-journal.org/sne-volumes/volume-33

SNE Basic e-Version with Open Access

SNE Full e-Version for publication-active EUROSIM Societies

A Simulator for NSA-DEVS in MATLAB.

D. Jammer, P. Junglas, T. Pawletta, S. Pawletta 141

Modeling and Simulation of a Real-world Application using NSA-DEVS. *D. Jammer, P. Junglas, T. Pawletta, S. Pawletta* 149

Fit for Duty Assessment of Driver Fatigue based on Statistical Modelling of Cardiovascular Parameters. *C. Pircher, M. Bachler, C. Ahlström, C. Mayer, B. Hametner* 157

Integrating Reinforcement Learning and Discrete Event Simulation Using the Concept of Experimental Frame: A Case Study with MATLAB/SimEvents. *T. Pawletta, J. Bartelt* . 167

Implementing Thermodynamic Cyclic Processes using the DLR ThermoFluid Stream Library. *P. Junglas* 175

Iterative Scenario-Based Testing in an Operational Design Domain for Artificial Intelligence Based Systems in Aviation. *B. Lukić, J. Sprockhoff, A. Ahlbrecht, S. Gupta, U. Durak* 183

Industrial Methods and Digital Tools for Craft Professionals – A Use Case Deploying Discrete Event Simulation and Virtual Reality. *B. Prell, J. Reiff-Stephan* 191

EUROSIM Societies & ARGESIM/SNE Short Info . N1 – N4 Conferences EUROSIM / ASIM Covers

Cover: Digital Marbling by Graham Horton, type ‘Serpentine’; digital-marbling.de

SNE Contact & Info

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

→ www.sne-journal.org

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

✉ SNE Editorial Office

Felix Breitenecker (Organisation, Author Mentoring)

Irmgard Husinsky (Web, Electronic Publishing),

Johannes Tanzler (Layout, Organisation),

ARGESIM/Math. Modelling & Simulation Group,

Inst. of Analysis and Scientific Computing, TU Wien

Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria

SNE SIMULATION NOTES EUROPE

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

Scope: Developments and trends in modelling and simulation

in various areas and in application and theory;

comparative studies and benchmarks (documentation 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, TU Wien, Math. Modelling Group

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

Print SNE: INTU (TU Wien), Wiedner Hauptstrasse 8-10,

1040, Vienna, Austria – www.intu.at

ARGESIM Publisher: ARBEITSGEMEINSCHAFT SIMULATION NEWS

c/o Math. Modelling and Simulation Group, TU Wien / 101,

Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria;

www.argesim.org, ✉ info@argesim.org on behalf of

ASIM www.asim-gi.org and EUROSIM → www.eurosims.info

© ARGESIM / EUROSIM / ASIM 2023

SNE - Aims and Scope

Simulation Notes Europe (SNE) provides an international, high-quality forum for presentation of new ideas and approaches in simulation - from modelling to experiment analysis, from implementation to verification, from validation to identification, from numerics to visualisation (www.sne-journal.org).

SNE seeks to serve scientists, researchers, developers and users of the simulation process across a variety of theoretical and applied fields in pursuit of novel ideas in simulation. SNE follows the recent developments and trends of modelling and simulation in new and/or joining areas, as complex systems and big data. SNE puts special emphasis on the overall view in simulation, and on comparative investigations, as benchmarks and comparisons in methodology and application. For this purpose, SNE documents the ARGESIM Benchmarks on *Modelling Approaches and Simulation Implementations* with publication of definitions, solutions and discussions. SNE welcomes also contributions in education in/for/with simulation.

SNE is the scientific membership journal of EUROSIM, the *Federation of European Simulation Societies and Simulation Groups* (www.eurosim.info), also providing Postconference publication for events of the member societies. SNE, primarily an electronic journal e-SNE (ISSN 2306-0271), follows an open access strategy, with free download in basic version (B/W, low resolution graphics). Members of most EUROSIM societies are entitled to download e-SNE in an elaborate full version (colour, high resolution graphics), and to access additional sources of benchmark publications, model sources, etc. (via group login of the society), print-SNE (ISSN 2305-9974) is available for specific groups of EUROSIM societies.

SNE is published by ARGESIM (www.argesim.org) on mandate of EUROSIM and ASIM (www.asim-gi.org), the German simulation society. SNE is DOI indexed with prefix 10.11128.

Author's Info. Individual submissions of scientific papers are welcome, as well as post-conference publications of contributions from conferences of EUROSIM societies. SNE welcomes special issues, either dedicated to special areas and/or new developments, or on occasion of events as conferences and workshops with special emphasis.

Authors are invited to submit contributions which have not been published and have not being considered for publication elsewhere to the SNE Editorial Office.

SNE distinguishes different types of contributions (*Notes*), i.e.

- TN Technical Note, 6–10 p.
- SN Short Note, max. 5 p.
- SW Software Note, 4–6 p.
- BN Benchmark Note, 2–10 p.
- ON Overview Note – only upon invitation, up to 14 p.
- EN Education Note, 6–8 p.
- PN Project Note 6–8 p.
- STN Student Note, 4–6 p., on supervisor's recommendation
- EBN Educational Benchmark Note, 4–10 p.

Further info and templates (doc, tex) at SNE's website, or from the Editor-in-Chief

www.sne-journal.org

office@sne-journal.org, eic@sne-journal.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 of submission to SNE (and extended for special issues and Postconference publication):

- Felix Breitenecker, Felix.Breitenecker@tuwien.ac.at
TU Wien, Math. Modelling, Austria, Editor-in-chief
- David Al-Dabass, david.al-dabass@ntu.ac.uk,
Nottingham Trent University, UK
- Maja Atanasijevic-Kunc, maja.atanasijevic@fe.uni-lj.si
Univ. of Ljubljana, Lab. Modelling & Control, Slovenia
- Aleš Belič, ales.belic@sandoz.com, Sandoz
- Peter Breedveld, P.C.Breedveld@el.utwente.nl
University of Twente, Netherlands
- Agostino Bruzzone, agostino@itim.unige.it
Universita degli Studi di Genova, Italy
- Vlatko Čerić, vceric@efzg.hr, Univ. Zagreb, Croatia
- Russell Cheng, rhc@maths.soton.ac.uk
University of Southampton, UK
- Roberto Cianci, cianci@dime.unige.it,
Math. Eng. and Simulation, Univ. Genova, Italy
- Eric Dahlquist, erik.dahlquist@mdh.se, Mälardalen Univ., Sweden
- Umut Durak, umut.durak@dlr.de
German Aerospace Center (DLR) Braunschweig, Germany
- Horst Ecker, Horst.Ecker@tuwien.ac.at
TU Wien, Inst. f. Mechanics, Austria
- Vadim Engelson, vadime@mathcore.com
MathCore Engineering, Linköping, Sweden
- Peter Groumpos, groumpos@ece.upatras.gr, Univ. of Patras, Greece
- Edmond Hajrizi, ehajrizi@ubt-uni.net
University for Business and Technology, Pristina, Kosovo
- Glenn Jenkins, GLJenkins@cardiff.ac.uk
Cardiff Metropolitan Univ., UK
- Emilio Jiménez, emilio.jimenez@unirioja.es
University of La Rioja, Spain
- Peter Junglas, peter@peter-junglas.de
Univ. PHTW Vechta, Mechatronics, Germany
- Esko Juuso, esko.juuso@oulu.fi
Univ. Oulu, Dept. Process/Environmental Eng., Finland
- Kaj Juslin, kaj.juslin@enbuscon.com, Enbuscon Ltd, Finland
- Andreas Körner, andreas.koerner@tuwien.ac.at
TU Wien, Math. E-Learning Dept., 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, thorsten.pawletta@hs-wismar.de
Univ. Wismar, Dept. Comp. Engineering, Wismar, Germany
- Niki Popper, niki.popper@dwh.at, dwh Simulation Services, 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
- Michal Štepanovský, stepami9@fit.cvut.cz
Technical Univ. Prague, Czech Republic
- Oliver Ullrich, oliver.ullrich@iais.fraunhofer.de
Fraunhofer IAIS, Germany
- 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
- Grégory Zacharewicz, gregory.zacharewicz@mines-ales.fr
IMT École des Mines d'Alès, France

A Simulator for NSA-DEVS in Matlab

David Jammer^{1,2*}, Peter Junglas², Thorsten Pawletta¹, Sven Pawletta¹

¹Research Group Computational Engineering and Automation, University of Applied Sciences Wismar, Philipp-Müller-Straße 14, 23966, Wismar, Germany; *david.jammer@cea-wismar.de

²PHWT-Institut, PHWT Vechta/Diepholz, Am Campus 2, 49356 Diepholz, Germany;

SNE 33(4), 2023, 141-148, DOI: 10.11128/sne.33.sw.10661
 Selected ASIM SST 2022 Postconf. Publication: 2023-08-15
 Rec. Revised Improved: 2023-11-27; Accepted: 2023-11-30
 SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
 ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. The PDEVS formalism is widely used for the description and analysis of discrete event systems. But PDEVS has some drawbacks in modeling Mealy behavior. A revised version (RPDEVS) has been invented to resolve them, but it has problems of its own, mainly because of its complicated simulator structure. The recently proposed NSA-DEVS scheme tries to unite the advantages of both formalisms by using infinitesimal time intervals.

To further substantiate this claim we describe an abstract simulator for NSA-DEVS, implement it in Matlab and simulate a simple queue-server system. This shows that NSA-DEVS combines the Mealy-like model description of RPDEVS with the simple simulator structure of PDEVS, making it a promising approach to implement an improved modeling and simulation system.

Introduction

The DEVS formalism [1] and its most popular variant PDEVS [2] are a well established approach for the modeling and analysis of discrete event systems. Although a few modeling and simulation tools exist that are using PDEVS [3], the usual formalism does not directly support the implementation of component-based simulation programs.

A few formal problems can be fixed by simple variations of the basic formalism [4], a well-known example being the introduction of input and output ports. A more serious flaw has been found by Preyser et al. [5]: Due to the Moore-like structure of PDEVS the combination of Mealy-type components can sometimes be difficult to implement.

Using the standard workaround of transitory states (i. e. states with transition times of 0) the behaviour of a complete system can always be modeled with PDEVS.

But the description of the underlying components as individual (“atomic”) blocks can lead to an ordering of concurrent events in the complete system, which does not agree with the intended behaviour.

Therefore Preyser et al. have introduced a Revised PDEVS (RPDEVS) formalism [6] that uses a Mealy-like scheme directly – without the introduction of transitory states – and allows for direct modeling of Mealy-like components, which behave correctly in the context of a larger system.

To make this possible they had to define an abstract simulator for RPDEVS [7] that uses a complicated scheme of internal iterations.

However, for systems with a complex causal structure of concurrent events this iteration leads to problems, as has been shown in [8] using the example of a queue-server system. To solve these problems and to bring the modeling process closer to the underlying ideas of the modeler, the NSA-DEVS (“Non-Standard Analysis DEVS”) formalism has been introduced in [8], which is a variant of RPDEVS and uses concepts of non-standard analysis [9].

Another approach has been suggested to cope with the ordering of concurrent events by augmenting the real time line [10], but it doesn’t address the Mealy-related problems that RPDEVS and NSA-DEVS try to solve.

The objective of the work presented here is to further investigate the soundness and usefulness of the NSA-DEVS formalism by defining a proper abstract simulator. To this end we first review the model and simulator specifications in PDEVS, then shortly introduce the hyperreal numbers and define the NSA-DEVS modeling formalism.

Next we describe the abstract NSA-DEVS simulator and highlight some crucial points of its implementation in Matlab. Finally we implement the queue-server system from [8] and demonstrate that it works as intended.

1 Short Review of the PDEVS Formalism

The Discrete Event System Specification (DEVS) formalisms are divided into model specification and abstract simulator which are explained in more detail in the following for Parallel DEVS (PDEVS). The model specification differentiates between atomic and coupled models, which together form a hierarchical structure. In the following, we introduce a simplified model specification for PDEVS that uses ports instead of input bags [11, p.108]. The model specification of an atomic model is an 8-tuple $\langle X, S, Y, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta \rangle$ with

X	set of input ports and values,
Y	set of output ports and values,
S	set of sequential states,
$\delta_{int} : S \rightarrow S$	internal transition function,
$\delta_{ext} : Q \times X^+ \rightarrow S$	external transition function,
$\delta_{con} : S \times X^+ \rightarrow S$	confluent transition function,
$\lambda : S \rightarrow Y^+$	output function,
$ta : S \rightarrow \mathbb{R}^{\geq 0} \cup \{\infty\}$	time advance function.

Here $Q = \{(s, e) | s \in S, 0 \leq e < ta(s)\}$ and e is the elapsed time since the last transition. The input and output sets are defined as

$$X = \{(p, v) | p \in P_{in}, v \in X_p\}$$

$$Y = \{(p, v) | p \in P_{out}, v \in Y_p\}$$

where P_{in} and P_{out} are the sets of input and output names and X_p and Y_p are the sets of possible values at input or output port p . Since inputs can arrive simultaneously at different ports, one needs the set

$$X^+ := \{(p_1, v_1), \dots, (p_n, v_n) | n \in \mathbb{N}_0, p_i \in P_{in}, p_i \neq p_j \text{ for } i \neq j, v_i \in X_{p_i}\}$$

and similarly Y^+ for simultaneous outputs at several ports. Unlike in [11] simultaneous inputs at the same port are not allowed here. This makes the formulation of external transition functions easier, but prohibits the direct connection of several output ports to one input port. This is not a real limitation though, since one can insert an appropriate atomic component (*multiplexer*) for this purpose.

The formal specification of coupled models has changed several times in the development of DEVS. For practical purposes, the following specification is used in this article: $N = \langle X, Y, D, \{M_d\}, EIC, EOC, IC \rangle$

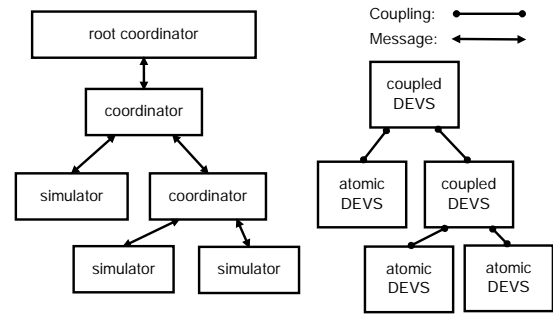


Figure 1: Hierarchical and distributed concept of the abstract simulator.

X	set of input ports and values,
Y	set of output ports and values,
D	set of component names,
$\{M_d\}$	set of dynamic systems with $d \in D$,
EIC	set of external input couplings,
EOC	set of external output couplings,
IC	set of internal couplings.

Furthermore, the PDEVS formalism defines an abstract simulator, which describes the execution of a specified model [11, p.197]. It consists of the modules *root coordinator*, *coordinator* and *simulator*. They are combined in a hierarchical structure, which is shown in Fig. 1 for a simple example.

The abstract simulator always consists of exactly one root coordinator as the topmost instance. This is always followed by a coordinator that is attached to the uppermost coupled model. In addition, a coordinator is assigned to each coupled model of the underlying layers of the hierarchical structure, whereas a simulator is assigned to each atomic model. The coordinators and simulators form a tree structure that parallels the model structure (cf. Fig. 1), where the leaves on the left side are the simulators and on the right (model) side the atomic components. The simulation is organized with a message concept. Messages are exchanged between root coordinator, coordinators and simulators, all downwards messages contain the current simulation time t . The following message types are used:

- i-message: downwards for initialization,
- *-message: downwards to initiate internal events,
- y-message: upwards to distribute outputs,
- x-message: downwards to trigger events,
- d-message: upwards to return information.

This terminology follows [7], in [11] the d-messages (“done”) are only implicitly mentioned in the pseudocode.

The simulation starts with an i-message that is sent by the root coordinator to the topmost coordinator and distributed downwards. Each simulator initialises its atomic model and returns the time of its next internal event to its parent coordinator. All coordinators collect the times of their children and report the smallest value upwards, until the root coordinator is reached, which stores the received value as the current simulation time.

Next the root-coordinator sends a *-message, which is forwarded according to the hierarchical structure to all simulators that are *imminent*, i. e. the time of their next event is equal to the current simulation time. Each of these simulators executes the λ -function of its atomic model and sends the output to its coordinator via a y-message. Using the set IC of its coupled model the coordinator distributes the outputs to the appropriate child simulators and coordinators via x-messages and sends additional empty x-messages to the imminent children. Furthermore the coordinator collects the external outputs according to the set EOC of its coupled model and sends them upwards via a y-message.

On receiving an x-message a simulator executes one of the three transition functions of its atomic model, depending on the event type. An empty x-message means an *internal* event, which causes the execution of the δ_{int} function. A non-empty x-message represents an *external* or *confluent* event. If the atomic model is not imminent, δ_{ext} is executed, otherwise δ_{conf} . After the execution of a transition function, the time advance function ta is called to compute the time of the next internal event, which is sent upwards. A coordinator, that receives an x-message, forwards it to its active children, i. e. those that get a new input or are imminent.

As a result, the root coordinator receives the next event time, updates the current simulation time and sends a new *-message. This procedure is repeated until the root coordinator detects a termination condition. A complete description of the abstract simulator using pseudocode is given in [11, p.350-353].

2 The NSA-DEVS Modeling Formalism

The basic idea of the NSA-DEVS formalism is to start with the RPDEVS description, to add infinitesimal delays at the inputs of all components and to replace tran-

sitory states by states with infinitesimal transition times. This has two immediate advantages: Firstly, the complex iteration, that is necessary in the RPDEVS simulator to handle the transport of events through networks of Mealy-type components, is obsolete. Secondly, one can easily define the ordering of concurrent events by using appropriate delay times.

The introduction of infinitesimals to represent small real delays avoids an abundance of unknown additional parameters. Instead one can mainly use a default value ϵ , using different values only for special needs. Furthermore, the simulator handles the infinitesimal events mainly internally, so that from the user perspective, correct Mealy behaviour can be achieved.

For a precise mathematical description of finite or infinitesimal time delays we use the totally ordered field of hyperreal numbers ${}^*\mathbb{R}$. This is an extension of the real numbers including an infinitesimal $\epsilon > 0$, which is smaller than any positive real number. Every finite hyperreal a is infinitely close to exactly one real number, called the standard part of a and denoted by $st(a)$. The construction of ${}^*\mathbb{R}$ relies on advanced results from set theory and logic, but its use is rather straightforward. Exact definitions, theorems and proves can be found in [9].

For the implementation of a simulator, numbers of the form $a + b\epsilon$ with $a, b \in \mathbb{R}$ are sufficient, they can be stored as a pair of floating point numbers. The standard part then simply is $st(a + b\epsilon) = a$. To represent *passive states*, i. e. states with an infinite transition time, the hyperreal number $\omega := 1/\epsilon$, represented by the floating point value “infinity”, can be used. ω is *unlimited*, i. e. it is larger than any real number. In the following we are mainly interested in the subset of positive finite hyperreals ${}^*\mathbb{R}_{fin}^{>0}$.

One can now formally define an *atomic NSA-DEVS* as a 7-tuple $\langle X, S, Y, \tau, ta, \delta, \lambda \rangle$ in the following way:

X	set of input ports and values,
S	set of states,
Y	set of output ports and values,
$\tau \in {}^*\mathbb{R}_{fin}^{>0}$	input delay time,
$ta : S \rightarrow {}^*\mathbb{R}_{fin}^{>0} \cup \{\omega\}$	time advance function,
$\delta : Q \times X^+ \rightarrow S$	transition function,
$\lambda : Q \times X^+ \rightarrow Y^+$	output function.

where the sets X, Y are defined as in section 1, but Q is changed slightly to $Q = \{(s, e) \mid s \in S, 0 < e \leq ta(s)\}$.

The main difference to the PDEVs formalism described above is the restriction to only one transition function and the extension of the output function, which is now called at all three kinds of events. This is identical to the RPDEVs definition in [6] and allows for a direct formulation of Mealy-type components. The formal difference to RPDEVs is small: All time values and intervals are now meant as subsets of the hyperreals ${}^*\mathbb{R}$ and ta is always > 0 . But the semantics are slightly different: When an external event, i.e. a set of inputs $x \in X^+$, occurs at time t , the output function λ is called at $t + \tau$, followed by an immediate call of δ . An internal event, i.e. a state change after a waiting time $ta(s)$, leads to a direct (undelayed) call of λ and δ . A concurrent incidence of a (delayed) external event and an internal event can be detected by both functions directly and doesn't need a special mechanism.

A *coupled NSA-DEVS* is defined just like in RPDEVs and PDEVs, outputs are transported as usual and a coupled component has no additional input delays. For the usual confirmation of closure under coupling, i. e. the formulation of a coupled system as an atomic component, one simply uses the smallest delay of all internal components that are connected to external inputs, and adds additional delays where necessary.

3 The Abstract NSA-DEVS Simulator

The general concept of the abstract simulator is the same as for PDEVs, it uses the hierarchical structure, the message system and the three modules that have been introduced in section 1. The algorithms of the coordinator and the root coordinator for NSA-DEVS are identical to the PDEVs versions described in [11, p.205] and [11, p.352-353], the only difference are the type of the current simulation time and the event times, which are now hyperreals instead of real numbers.

The basic difference lies in the algorithm of the simulator module: Though it looks similar to the PDEVs simulator in [11, p.351], it implements the NSA-DEVS scheme, which directly supports Mealy-like behaviour using infinitesimal input delays. Its simplified pseudocode description is presented in Listing 1.

Lines 1–7 list the variables used by the simulator. The first five are the same as for PDEVs: the parent coordinator, the times of the last and the next event, the attached model – with the atomic NSA-DEVS structure and its complete state – and the output values.

Listing 1: NSA-DEVS Simulator.

```

1 properties:
2   parent
3   t1
4   tn
5   model (NSA-DEVS incl.  $\tau$  and
           total state (s,e))
6   y
7   x*
8
9 when receive i-message(i,t) at time t
10  t1 = t - e
11  tn = t1 + ta(s)
12
13 when receive *-message(*,t) at time t
14  e = t - t1
15  y =  $\lambda(s, e, x^*)$ 
16
17  send y-message(y,t) to parent
18  coordinator
19
20 when receive x-message(x,t) at time t
21  if x ==  $\emptyset$ 
22    e = t - t1
23    s =  $\delta(s, e, x^*)$ 
24    x* =  $\emptyset$ 
25    if ta(s) ==  $\omega$ 
26      tn =  $\omega$ 
27    else if st(ta(s)) == 0
28      tn = t + ta(s)
29    else
30      tn = st(t + ta(s))
31  t1 = t;
32 else
33   if not (x* ==  $\emptyset$ )
34     add events from x to x*
35   else
36     x* = x
37     tn = t +  $\tau$ 

```

Since the input delay is realized inside the simulator, input values must be stored temporarily, using the variable x^* . In lines 9–11 the i-message is handled, which just computes the times of the last and the next events. The *-message is processed in lines 13–18, where the elapsed time is calculated, the λ function is executed and the y-message is sent to the parent coordinator. In contrast to the PDEVs algorithm λ is now a function of the total state (s,e) and the input value x^* that has been stored before.

The new part - compared to PDEVs - is the way the x-message is processed, which is shown in lines 19–36. It discriminates between an internal event (lines 21–30, $x == \emptyset$) and an external event (lines 32–36). In the latter case it stores all incoming values in x^* and schedules a new internal event at the delayed time $t + \tau$.

All internal events are handled by calling the transition function δ and computing the time of the next event using ta in lines 24–29. This calculation deserves special attention: It guarantees that “real” time steps (i. e. non-infinitesimal ones) lead to real valued time values in order to implement a correct timing and to prohibit an accumulation of infinitesimal delays.

To better understand the operation of the abstract simulator, especially how it creates a proper Mealy behaviour, we introduce a simple example model N: It consists of a generator G, which outputs a value $t/10$ at times $t = 1, 2, 3 \dots$, a multiplication block M, which multiplies its input by a factor 3, and a terminator component T, which acts as a sink for the incoming values (cf. Fig. 2).

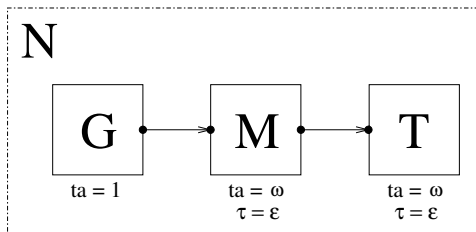


Figure 2: Simple example model.

The messages that are sent between the root coordinator RC, the coordinator C_N of coupled model N and the three simulators S_G , S_M and S_T with the associated models G, M and T are shown as a sequence diagram in Fig. 3. Downwards messages are denoted as $(msg\ type, current\ time)$, for the x-messages the input value is added. Upwards messages are shown as $(msg\ type, result)$.

In the initialisation step at $t = 0$, an i-message is sent and distributed to the simulators, returning the time $t = 1$ of the first internal event to RC. This is followed at $t = 1$ by a *-message sent to the simulator of the only imminent component G, which generates an output event and sends a y-message with its output 0.1 back to C_N . The coordinator now sends an empty x-message to S_G , which returns the time $t = 2$ of its next internal event. Moreover, C_N sends a non-empty x-message to S_M , which stores the value internally and schedules a new internal event according to the input delay time.

The next *-message at $t = 1 + \epsilon$ arrives at the simulator of the imminent component M, which calls its λ function and sends the output value 0.3 as a y-message to its coordinator.

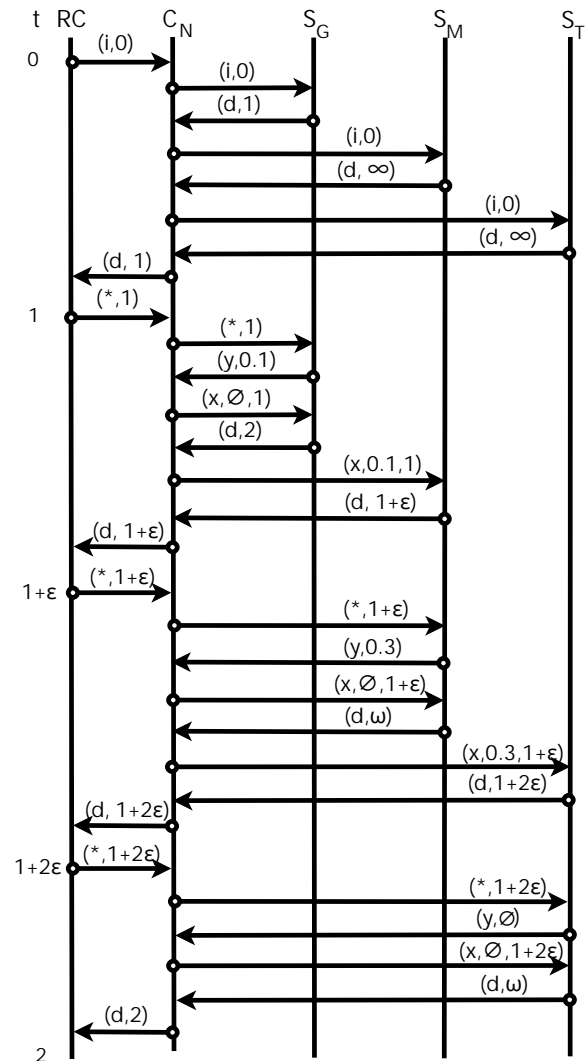


Figure 3: Example of message flow.

C_N now sends an empty x-message to S_M , which calls its δ and ta functions and returns $t = \omega$ to C_N . M is now in a passive state. The rest of the diagram shows how the output value propagates to S_T , which just terminates the incoming events. Since the coordinator stores all future event times of its children, it finally returns $t = 2$ (originally coming from S_G) as the time of the next event, which will repeat the whole cycle.

This example shows precisely, how the two parts of the x-message algorithm in the simulator module work together to implement the time delay and the Mealy behaviour of a simple Mealy block such as a multiplication function.

4 Implementation of the NSA-DEVS Simulator

The “infinitesimal cloud” of hyperreal numbers around each real number has a complex structure with lots of different layers of smallness, e. g. using ϵ^2 or $\sqrt{\epsilon}$. The purpose of using $\ast\mathbb{R}$ in the formulation of NSA-DEVS is the possibility to introduce short time intervals without defining their size explicitly, but still being able to order them.

Therefore times of the form $a + b\epsilon$ are sufficient here, they are stored as two-element vectors. The implementation of time comparisons and sorting has to be adapted accordingly.

Since the goal of NSA-DEVS is to provide a good basis for the concrete modeling of discrete event systems, an implementation should free the modeler from the tedious task of defining lots of additional infinitesimal parameters. Therefore the concrete simulator contains a variable $\tau_{def} = r\epsilon$ (usually $r = 1$) that is used as a default value of τ for all atomic components. Furthermore the user can still define transitory states with a transition time $ta(s) = 0$, which is replaced automatically by setting $ta(s) = r\epsilon$.

For debugging purposes it would be useful to make the infinitesimal delays explicitly visible. To this end the simulator contains a real (i. e. floating point) parameter μ , which is 0 normally, but can be set to a value larger than zero for debugging.

In this case the infinitesimal ϵ is replaced by μ and all times are real values computed as

$$t' = \begin{cases} (t(1), t(2)) & \text{if } \mu = 0, \\ (t(1) + \mu t(2), 0) & \text{if } \mu > 0. \end{cases}$$

In complex models the value of μ has to be chosen carefully: It should be large enough to make the infinitesimal internal processes visible, but small enough to not induce any changes into the behaviour of the model. As a result of this extension, the implementation of the x-message gets more complicated, as can be seen in Listing 2.

The special case of passive states in Listing 1 (l. 24f) is done automatically in line 16 due to the handling of the value “infinity” in floating-point arithmetic.

Listing 2: Implementation of the x-message algorithm.

```

1 when receive x-message(x,t) at time t
2   if x == ∅
3     e = [t(1) - t1(1), t(2) - t1(2)]
4     s = δ(s,e,x*)
5     x* = ∅
6
7     tb = ta(s)
8     if tb == [0,0]
9       tb = [0, r]
10    if tb(1) == 0
11      if μ == 0
12        tn = [t(1), t(2) + tb(2)]
13      else
14        tn = [t(1) + μ*tb(2), 0]
15    else
16      tn = [t(1) + tb(1), 0]
17    t1 = t;
18  else
19    if not (x* == ∅)
20      add events from x to x*
21    else
22      x* = x
23    if μ == 0
24      tn = [t(1) + tau(1), t(2) + tau(2)]
25    else
26      tn = [t(1) + tau(1) + μ*tau(2), 0]
    
```

5 Case Study: A Simple Queue-Server System

To test the operation of the complete NSA-DEVS formalism – model specification and simulator –, a prototype has been created in Matlab, which is named NSA-DEVSforMATLAB.

It contains all features introduced in section 4 and is implemented in an object-oriented way. To show its functionality, the singleserver example from [8] has been chosen as an example for this article; it is shown as a block diagram in Figure 4.

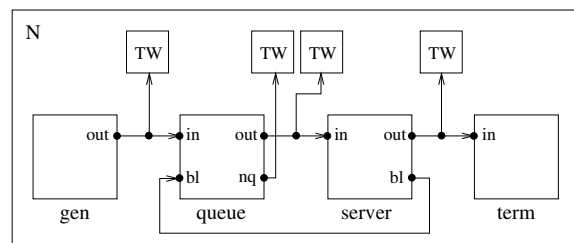


Figure 4: Example model singleserver.

The following atomic models are used for the example:

- Generator: produces entities with an interval of one second,
- Queue: infinite queue,
- Server: service time 1.5 s,
- Terminator: terminates the entities,
- ToWorkspace (TW): logging data.

The special feature of this model is that the queue should only send entities to the server when it is not busy. The server announces this information via port blocked (bl), which is sent to port bl of the queue. For functionality, the input delay of the queue must be greater than the input delay of the server. The input delays of the generator and terminator do not matter. A special role is played by the four ToWorkspace models, which are connected to the output ports *out* and *nq* according to Fig. 4 and store the output values. They too – like every atomic model – have an input delay.

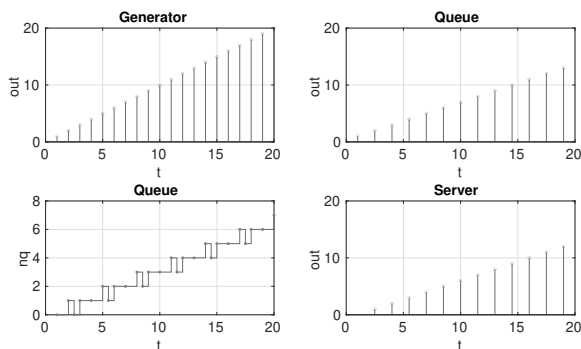


Figure 5: Simulation results with high input delay at ToWorkspace.

For the first simulation run, a large infinitesimal input delay was chosen for the ToWorkspace models. The result can be seen in Figure 5, which displays generator output (top left), queue output (top right), queue load (bottom left) and server output (bottom right).

The high input delay has the effect that output changes, which happen during a series of infinitesimal time intervals, are discarded inside a ToWorkspace block and only the final value before a finite time step is shown. This can be seen, for example, at time 10: The server has finished processing and is idle. Therefore the queue sends an entity to the server.

At the same time, the generator also outputs an entity and sends it to the queue. In total, the load of the queue does not change.

However, if one uses a low input delay for the ToWorkspace models, one sees that the queue load at time 10 has the values 3 and 4 simultaneously. This means that the new entity enters the queue first and then an entity is sent to the server. This behavior is shown in Figure 6. One could use the debug mode, i. e. set the parameter μ to a finite value, to dissolve the “spike” at $t=10$ into a small step, thereby clearly showing the internal ordering of the events.

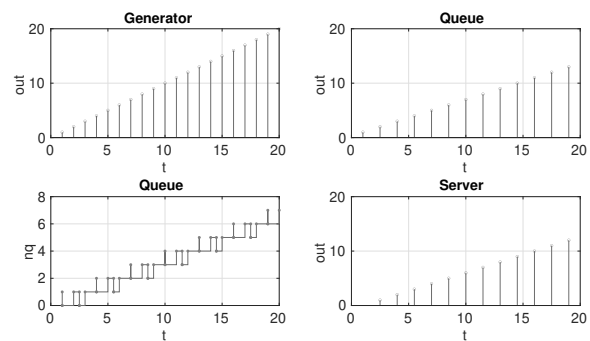


Figure 6: Simulation results with low input delay at ToWorkspace.

6 Conclusion

With the specification of an abstract simulator, which defines the behaviour of a model consisting of atomic and coupled components, the description of the NSA-DEVS formalism is now formally complete.

We have shown that NSA-DEVS is able to directly describe Mealy-like models in the same way as RPDEVs, but with a much simpler simulator algorithm similar to the original PDEVs version. In this way NSA-DEVS combines the best of the two preceding formalisms.

Its principle usability has been demonstrated by the implementation of the simulator and a non-trivial example model in Matlab. The notoriously difficult modeling of concurrent events has been substituted by a clear definition of an ordering based on infinitesimal delays.

The inclusion of a debugging mode further helps to understand the corresponding difficulties. An interesting side effect is the possibility to easily model systems with finite time delays.

At first sight, the NSA-DEVS approach seems to be very similar to the concept of superdense time [10], where a real time value is augmented by a natural number to order concurrent events. But the much richer structure of $\ast\mathbb{R}$ – even of the small part that is used in the implementation – has profound consequences: On the practical side, one can use real infinitesimal delays to squeeze an event between existing ones, without the need to reorder the complete sequence.

The conceptual difference, however, is the dynamic structure of NSA-DEVS: The order of concurrent events is defined by the infinitesimal delays in the complete model, which add up in a “realistic” way. While the fixed ordering of superdense time is similar to the *Select* function in Classical DEVS [11, p.104], NSA-DEVS – like PDEVS – allows concurrent events on the infinitesimal scale and parallelism.

To further examine the practical usefulness of the NSA-DEVS formalism, one should next study a set of standard examples with complex event cascades and real-world case studies. This could help answering the crucial question, whether an abundance of new parameters is necessary in real models or if the use of a default delay is sufficient in many cases. Another interesting question is, whether one delay for an atomic model suffices, or if one needs port specific delay times.

Finally one should address the practical usefulness of the simulator and its implementation: How does it perform in comparison to existing PDEVS or RPDEVS simulators? Suitable benchmarks would address simulation times as well as the number of internal messages used inside a simulator.

Although the definition of the simulator is a large step forward, much remains to be done before NSA-DEVS can be considered a solid approach for practical discrete event modeling and simulation.

References

- [1] Zeigler BP. *Theory of Modeling and Simulation*. New York: Wiley-Interscience, 1st ed. 1976.
- [2] Chow ACH. Parallel DEVS: A Parallel, Hierarchical, Modular Modeling Formalism and its Distributed Simulators. *Transactions of The Society for Computer Simulation International*. 1996;13(2):55–67.
- [3] Franceschini R, Bisgambiglia PA, Touraille L, Bisgambiglia P, Hill D. A survey of modelling and simulation software frameworks using Discrete Event System Specification. In: *Proc. of 2014 Imperial College Computing Student Workshop*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2014; pp. 40–49.
- [4] Goldstein R, Breslav S, Khan A. Informal DEVS conventions motivated by practical considerations. In: *Proc. of Symposium on Theory of Modeling & Simulation – DEVS Integrative M&S Symposium*. 2013; pp. 10:1–10:6.
- [5] Preyser FJ, Heinzl B, Raich P, Kastner W. Towards Extending the Parallel-DEVS Formalism to Improve Component Modularity. In: *Proc. of ASIM-Workshop STS/GMMS*. Lippstadt. 2016; pp. 83–89.
- [6] Preyser FJ, Heinzl B, Kastner W. RPDEVS: Revising the Parallel Discrete Event System Specification. In: *9th Vienna Int. Conf. Mathematical Modelling*. Wien. 2018; pp. 242–247.
- [7] Preyser FJ, Heinzl B, Kastner W. RPDEVS Abstract Simulator. *SNE Simulation Notes Europe*. 2019; 29(2):79–84. doi: 10.11128/sne.29.tn.10473.
- [8] Junglas P. NSA-DEVS: Combining Mealy Behaviour and Causality. *SNE Simulation Notes Europe*. 2021; 31(2):73–80. doi: 10.11128/sne.31.tn.10564.
- [9] Goldblatt R. *Lectures on the Hyperreals*. New York: Springer. 1998.
- [10] Sarjoughian HS, Sundaramoorthi S. Superdense time trajectories for DEVS simulation models. In: *SpringSim (TMS-DEVS)*. 2015; pp. 249–256.
- [11] Zeigler BP, Muzy A, Kofman E. *Theory of Modeling and Simulation*. San Diego: Academic Press, 3rd ed. 2019.

Modeling and Simulation of a Real-world Application using NSA-DEVS

David Jammer^{1,2}, Peter Junglas^{2*}, Thorsten Pawletta¹, Sven Pawletta¹

¹Research Group Computational Engineering and Automation, University of Applied Sciences Wismar, Philipp-Müller-Straße 14, 23966, Wismar, Germany;

²PHWT-Institut, PHWT Vechta/Diepholz, Am Campus 2, 49356 Diepholz, Germany; *peter@peter-junglas.de

SNE 33(4), 2023, 149-156, DOI: 10.11128/sne.33.tn.10662
 Received: 2023-11-03; Revised: 2023-11-25
 Accepted: 2023-11-26
 SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
 ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. The recently proposed NSA-DEVS formalism uses infinitesimal time delays to combine the easy implementation of Mealy components from RPDEVS with the simple simulator structure of PDEVS in order to make DEVS a suitable foundation for complex component-based modeling tasks. To prove its general applicability, it is used here to implement a large real-world model describing a production line that consists of several furnaces, lathes and grinders. Using a Matlab implementation of the simulator algorithm, the model behaviour is analyzed carefully and it is found that only one of its over 400 infinitesimal time delay parameters has to be changed from its default value. This shows the soundness of the basic ideas of NSA-DEVS and its applicability for real-world examples.

Introduction

A well-known difficulty that often arises when using the discrete event approach is the occurrence of events at the same time instant. There are two very different reasons for such a behaviour: The first one is the *accidental coincidence* of events, e. g. when two input events from different sources arrive at the same time, or when an event arrives at a component at the exact moment of a state change. In such cases the exact ordering of these events inside a concrete simulator is often of no consequence, therefore providing an opportunity for parallel execution. The other reason is a chain of events that is created by an initial event, causing an immediate (*transitory*) state change that leads to further events that

spread through the system without delay. Even if all these events formally appear at the same time instant, their logical relation enforces a fixed ordering of such a *causal cascade*.

In the context of the widely used DEVS formalism [1] the behaviour of a model and its simulator are defined precisely. Different variants of the formalism provide specific mechanisms to fix the order of concurrent events, where necessary: Classic DEVS uses a *Select* function on the level of coupled components, while PDEVS introduces a *confluent state transition* function inside an atomic component. The revised PDEVS formalism (*RPDEVS*) [2] incorporates a direct Mealy structure with a generic state transition function and a refined simulator algorithm [3].

In order to simplify the modeling of causal cascades and the modeling of Mealy behavior, a new approach named NSA-DEVS (*Non-Standard Analysis DEVS*) was proposed in [4]. It is based on the physical intuition that the transport and the processing of events always need a certain amount of time, so that the simulation problems are actually due to oversimplification. But instead of introducing a plethora of small delay parameters, NSA-DEVS uses infinitesimal time delays. Therefore it defines time values as elements of the hyperreals ${}^*\mathbb{R}$, which is a mathematically well-defined field extension of \mathbb{R} containing an infinitesimal $\varepsilon > 0$ [5].

The definition of a simulator [6] and the careful analysis of a set of standard examples [7] have shown the basic soundness of the new formalism. But to prove its general applicability, one has to test it with a large non-trivial example. For this purpose a model of a production line will be studied in the following that consists of several furnaces, lathes and grinders. The model includes the material flow, several process controllers and basic physical properties.

After a short recapitulation of the NSA-DEVS formalism, the model and its implementation in Matlab will be described, highlighting some special atomic and coupled components. Finally, the simulation of the model and careful analysis of the results will clarify the main question: How many of the delay parameters have to be changed from their default values, and how difficult is it to find them? Or, in other words: Is the NSA-DEVS formalism suitable for real-world applications?

1 The NSA-DEVS Formalism

For the convenience of the reader the basic definitions of the NSA-DEVS model specification will be given here. A more detailed description, the connection with the PDEVS and RPDEVS formalisms and the definition of the abstract simulator can be found in [6].

Two types of models are defined in all DEVS variants: an atomic model that describes the behaviour of a single component, and a coupled model, which specifies how models are combined to build a hierarchical structure. In NSA-DEVS an atomic model is given by a 7-tuple $\langle X, S, Y, \tau, ta, \delta, \lambda \rangle$ with

X	set of input ports and values,
S	set of states,
Y	set of output ports and values,
$\tau \in {}^*\mathbb{R}_{fin}^{>0}$	input delay time,
$ta : S \rightarrow {}^*\mathbb{R}_{fin}^{>0} \cup \{\omega\}$	time advance function,
$\delta : Q \times X^+ \rightarrow S$	state transition function,
$\lambda : Q \times X^+ \rightarrow Y^+$	output function.

The input and output sets X, Y contain pairs of ports and values, where ports are given by names. The sets X^+, Y^+ consist of sets of values from X, Y to describe the simultaneous appearance of input or output values at different ports. The definition of the transition function δ and the output function λ contain the set $Q = \{(s, e) | s \in S, 0 < e \leq ta(s)\}$ that combines a state and the elapsed time e since the last transition.

As in RPDEVS, both event types (incoming event or internal event) lead to a call of λ followed by a change to a new state according to δ . The time advance function ta may be infinitesimal or infinite (using $\omega := 1/\varepsilon$, with ε as infinitesimal value), but it is always > 0 , thereby excluding proper transitory states. The delay time τ between the arrival of a set of inputs and the call of λ and δ is generally an infinitesimal, often given by a default value $\tau_{def} = \varepsilon$.

A coupled NSA-DEVS model is defined as in PDEVS and RPDEVS. It consists of input and output ports and a set of atomic or coupled models, which are connected among themselves and to the external ports. Outputs are transported as usual and a coupled component has no additional input delays.

2 A Real-world DEVS Application

The model and its components described in this article were developed similar to [8, 9]. The model describes a production line which includes lathes, grinders and furnaces (cf. Figure 1). In this production line, in the first step, the raw parts supplied by a generator are processed by lathes. After lathing, the components are thermally treated. This is done first by a furnace for volume hardening (*furnace 1*) and then by a furnace for stress relief annealing (*furnace 2*). The two furnaces have no difference in design, but only in their temperature behaviour. The furnaces always process several parts at the same time, depending on the type of furnace. After heat processing, the components are finished by grinders. The manufacturing operations are decoupled via buffers with a maximum capacity of 400 parts. The buffers of the lathes and grinders are located in the coupled model of the respective manufacturing operation (cf. Figure 2). The buffers are organized in a coupled model whereby every single machine has its own buffer (cf. Figure 3). This model corresponds to example 4 from [7].

The lathe, grinder and furnace machine models have all the same internal structure according to [10]. The structure is divided into a physical model (PM), control model (CM) and material flow model (MF) (cf. Figure 4). The PM describes the physical relationships, e.g. the heat flows, by differential equations. The CM contains the local machine control, which takes into account various internal processing steps. In MF, the internal material flow is modeled in a process-oriented way, describing the parts as moving entities.

The machine models return the process variables electrical power, electrical energy and utilization over time. The buffers deliver information about the current load and a terminator reports the number of finished parts. These process variables are used to calculate the Key Performance Indicator (KPI) variables buffer stock (*pcStock*), throughput (*thrput*), load peak (*loadPeak*), utilization (*pcUtil*), energy per part (*eSpec*) and production time per part (*procTime*).

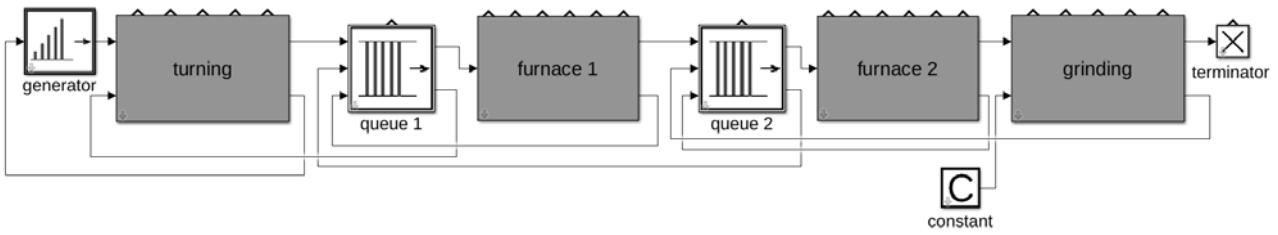


Figure 1: Top level structure of the production line.

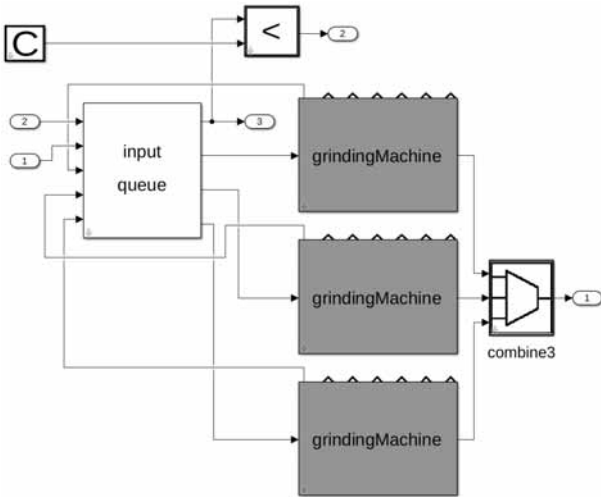


Figure 2: Substructure of the grinding component in Fig. 1.

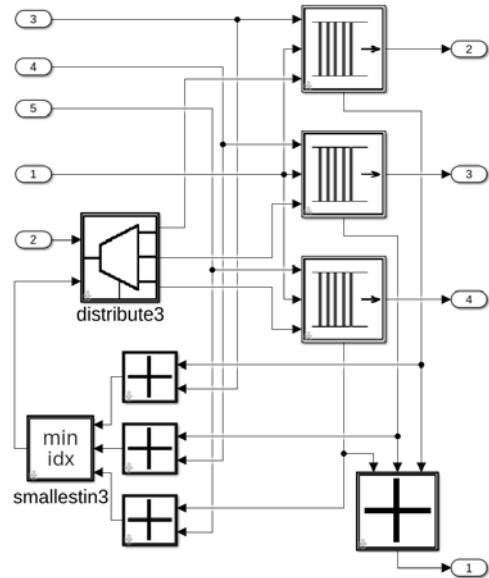


Figure 3: Substructure of the input queue in Fig. 2.

The entire model consists of 391 atomic models and 88 coupled models and is organized in 5 hierarchical levels.

3 Atomic Components

Using the Matlab-based NSA-DEVS simulator from [11], an atomic model is implemented as a class, which defines the transition and the output function as methods $\delta(obj, e, x)$ and $\lambda(obj, e, x)$ and the time advance function as method $ta(obj)$. The argument obj is the handle (reference) to the object, allowing access to its state, e is the time since the last transition, given as a two-dimensional vector $[a, b]$ denoting a hyperreal value $a + b\epsilon$, and x is a structure, containing the current input as field/value pairs, where the field is identical to the port name. The constructor always defines the name of the component, the input delay τ and a debug flag, as well as optional model parameters.

Since NSA-DEVS retains the mealy-type behaviour of RPDEVS, standard computational components can be implemented easily. The main difference to components used in a continuous modeling environment is due to the event-based paradigm applied here: At least for components with more than one input port, one needs a state variable for every input port to store incoming values, because an input is only defined at the time of the corresponding input event. Taking this into account, one can implement a simple adding component by providing a δ function that just stores the input values, a ta function that always returns $[inf, 0]$, and a λ function that returns the sum of the input values, using the stored values, where necessary.

A standard library of atomic components has been built for the models described in [7] and the production line, containing mathematical and routing components, simple source and sink components and several logistics related components such as a queue and a server.

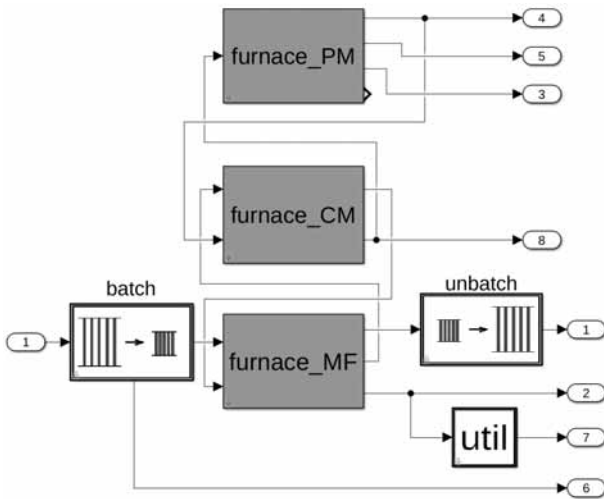


Figure 4: Substructure of the furnace components in Fig. 1.

Several of them have been described in [7], among them an atomic model `ToWorkspace` that can be connected to an output port and copies the incoming values to a global output variable, which can be analysed after the simulation run. For the modeling of the furnace a batch component has been added that combines a given number of incoming entities into one batch entity, together with a corresponding unbatch atomic model.

Since some physical variables of the production line are modelled by differential equations, e. g. the temperature inside a furnace or the electric current in a grinding machine, an integrator component is needed inside the NSA-DEVS environment. For this purpose the Quantized-State-Systems (QSS) method is used [12], which has already been implemented inside a Matlab-based DEVS simulator [13].

In addition, a few atomics have been added that are specific to the production line: a special server component, controller components for all machine types and a few convenience components, which could have been built as coupled models from atomics in the standard library. Altogether the production line model uses 15 types of atomics from the standard library and 7 from its own library.

4 Coupled Components

In the NSA-DEVS simulator used here, a coupled model is not implemented as a class, but simply given by a constructor function, which defines all elements

of the coupled-model specification: its internal atomic and coupled models and their connections among themselves and to the external ports. Furthermore, it assigns its atomic models to simulator modules and its coupled models – including the currently defined one – to coordinator modules (cf. [6]).

Especially for large coupled models, the programming of such a constructor function is tedious and error prone. Therefore a graphical model generator is provided – similar to the approach in [14] – that creates the function from a graphical description. For this purpose, atomic models are represented in Simulink libraries as blocks that only contain the external ports, using masks to define their parameters (cf. Figure 5). Coupled models can then be defined as standard Simulink subsystems consisting entirely of such blocks, subsystems and external ports. The model generator creates all needed constructor functions, where the top-level model can be directly run in the simulator. The corresponding model of the production line is shown in Figure 1. It consists of three atomics – a generator, a constant and a terminator – and coupled systems for the furnaces, lathes, grinders and intermediate queues.

At the inner hierarchy levels many models look very much like similar models in continuous simulation environments like Simulink. A good example is the component that computes the furnace temperature using the simple differential equation

$$C_O \frac{dT}{dt} = k_A(T - T_e) + P_{heating} - P_{unload} .$$

It is built exactly like a corresponding Simulink model (cf. Figure 6).

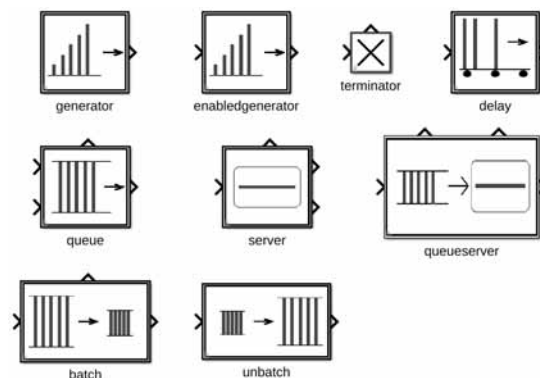


Figure 5: Library of logistics related components.

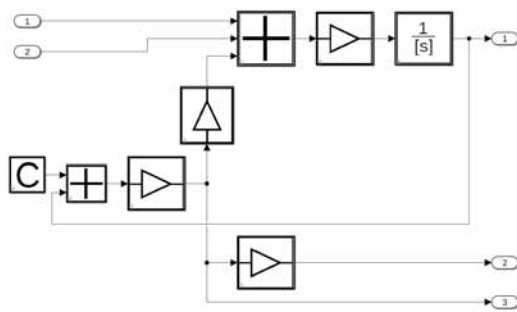


Figure 6: Model for the computation of the furnace temperature.

Though this correspondence may sometimes be helpful for the definition of suitable atomics and the construction of coupled models, it is basically superficial. The main difference is, of course, the meaning of the connecting lines: In continuous simulation environments a line transports a signal that has always a value, while in a DEVS environment a value is only defined at the moment of an input event. This has practical consequences for the concrete modeling, e. g. for the question, whether it is possible to connect an output port directly to several input ports (*1:N connection*) or vice versa (*N:1 connection*).

Using a signal paradigm as in Simulink, the 1:N direction clearly is possible, distributing the value to all input ports, while the reverse direction is incorrect, since in this case the input value is not well defined. In PDEVS both directions are allowed: Several events can arrive at the same input port, even simultaneously, while an output event is copied automatically by the coordinator, when it is distributed to several components. In NSA-DEVS the situation is different again: While N:1 connections are possible in principle, the abstract simulator does not support the simultaneous arrival of input events at the same port. If this is required – e. g. for modeling of multi-value logic components [15] – one has to use corresponding connection atomics explicitly. The 1:N direction is handled by the coordinator as in PDEVS and will often be used, e. g. to attach `ToWorkspace` components directly to a line or to distribute events transporting simple values. If the data of an event is interpreted as an entity, as in transactional-based modeling, it would be better though to include explicit copy components to make the intention clear.

5 Testing and Running the Model

We will now try to run the complete production line model and interpret the simulation results. The main point of interest here is, how to set all the infinitesimal parameters. They consist primarily of the 391 input delay times τ . Furthermore one needs 12 additional delay parameters for the transitory states that are used in the queue, unbatch and combine atomics, which will be denoted as τ_D . All these values are usually predefined and set to the value $\tau_{def} = \varepsilon$. From the analysis of a few example models in [7] the following situations have been identified, where one has to change some of the parameters from their default value:

- a.) Input events that appear during the input delay time of a component, overwrite a previous input value at the same port, which sometimes is useful, but more often not. A particular example is a combine component that serializes concurrent incoming inputs: These should be output with a sufficiently large delay time, so that subsequent components can process them one after the other.
- b.) To make a queue-server combination work, the blocking signal from the server has to arrive before a second entity is output by the queue. In standard situations it is sufficient to set the delay time of the “transitory” queue state to 2ε , therefore this value is defined as default in the library queue block.
- c.) In loops containing several sequences of components the order of concurrent events depends on the total delay times along different paths. If one wants to implement a specific ordering, one can slow down some paths by increasing appropriate delays.

Since problems due to wrong delay values often lead to missing entities, a simple test strategy is to insert a fixed amount of input entities and run the model, until all entities should have reached the output. We start with default parameters, i. e. all τ and τ_D values are set to ε , except the τ_D values of queue atomics, which are set to 2ε . The simulation run shows that no entities reach the final terminator and that the total amount `pcStock` of entities in internal queues goes down to zero, i. e. all entities are lost.

Taking into account the lesson from a.) the culprits are quickly identified as the unbatch and combine atomics: They release entities in groups with only the default

delay in between, so that these entities overwrite each other during the input delay of the following component. An obvious solution seems to be to enlarge the τ_D values of both `unbatch` and both `combine` atomics to 2ϵ . Running the modified model, the simulator hangs and must be terminated manually. This can happen, when the overwriting of inputs happens in a loop without reaching a result.

To localize the problem, one has to look more closely at the sequence of events using the debugging possibilities of the simulator. Debugging a complex discrete-event based application is always a difficult endeavour. Therefore one usually starts by reducing the complexity and looking at test benches for basic subsystems. Though this approach has been followed here to find the usual bugs, it is not sufficient to fix all τ values, because some important loops only show up in the complete model.

When setting a debug flag, the simulator outputs the current simulation time. This shows that the simulation is stuck in a loop at the first time, when a complete batch of entities leaves the second furnace (Figure 4) and enters the grinding section (Figure 2). The critical region therefore seems to be the `unbatch` atomic inside furnace 2 and the `input_queue` component (Figure 3) inside grinding. Figure 8 shows this model part with a view of the internal structure of the coupled subsystems. Guessing from previous experience, one would suspect that the τ_D parameter of `unbatch` is too small. With a value of 10ϵ for the `unbatch` components in both furnace subsystems, the model works, no entities are lost, the output is as expected (Figure 7).

Since the problem encountered is typical for the behaviour of NSA-DEVS, we will show in detail, how one can use the debugging features of the simulator to find the concrete source of the error (cf. Figure 8). First, one adds `toWorkspace` components inside `input_queue` that show the number of entities in the three queues and the ids of outgoing entities. Unfortunately, their results are not directly available, when the simulation run is interrupted manually. Here, another debugging feature is useful: All atomics have debug flags, which can be switched on individually to create outputs of their input and output values and all state changes during the simulation run. This feature is used here for the six `toWorkspace` components and the `distribute3` component at the input of the `input_queue` coupling.

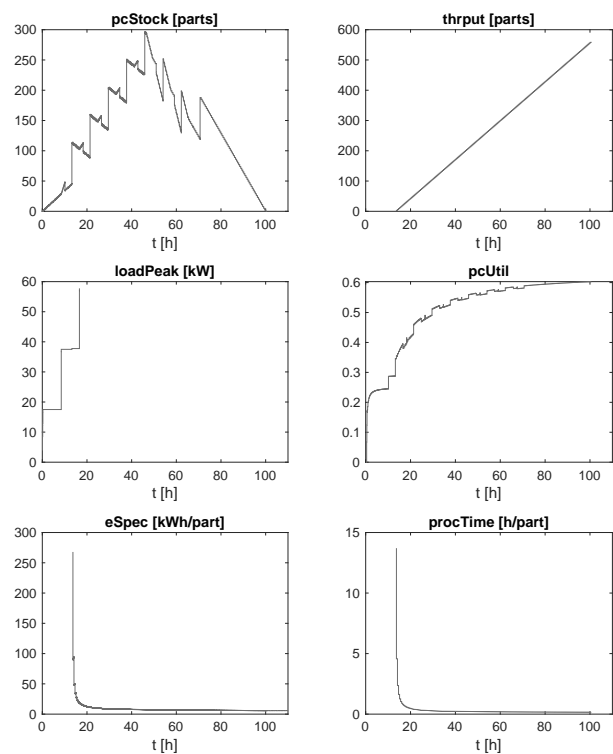


Figure 7: Simulation results showing KPI variables of the production line model.

Analyzing the debugging outputs of the working version, which uses $\tau_D = 10\epsilon$ for the `unbatch` component, shows the following behaviour: The first three entities are distributed to the three queues, leave their queue immediately and enter the server inside the corresponding grinding machine. This leads to several changes of the `port` input of `distribute3`, which points to the currently shortest queue/server line. The input events at `port` are delayed, because the event cascade has to pass the queue, server, add and `smallestIn` atomics. Some of them are overwritten by the following ones, as can easily be seen in the debugging log: In such a case the lambda function of a component is called several times without an intervening call to the delta function. But this only effects intermediate values here and doesn't lead to erroneous behaviour. The next entities (no. 4, 5, ...) are stored in the queues, the associated new `port` values arrive and the `distribute3` component is ready each time, before the next entity enters the `input_queue`, due to the long τ_D -delay inside `unbatch`.

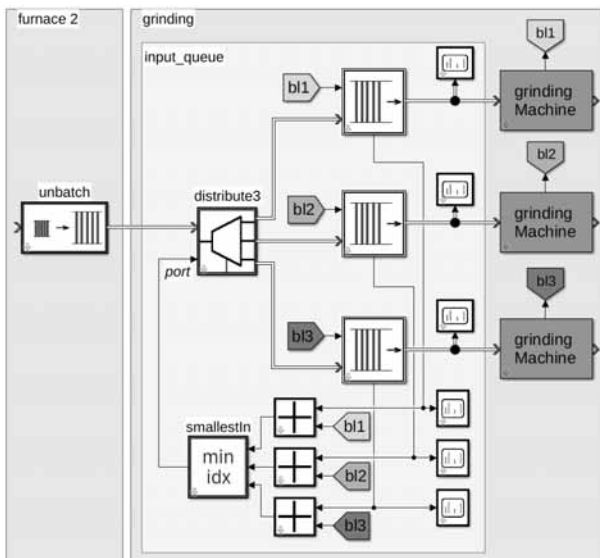


Figure 8: Critical part of the production line model.

The previously considered version with the smaller value of $\tau_D = 2\epsilon$ in `unbatch` shows a completely different sequence of actions: The first two entities are distributed to queue 1, the third to queue 2, since these entities arrive before the `port` input could adapt to the changes.

After the first entities the queues are blocked. This reduces the delay chain of the `port` signal, because the servers inside the grinding machines are no longer involved, with drastic consequences: The new entities now arrive at the same time as new `port` values from the previous entity. Therefore the old values are overwritten and new lambda calls scheduled, without any intervening delta calls.

This leads to a loss of all entities, until the last of the batch arrives. But still the new `port` numbers come in, therefore the loop doesn't end even now.

This behaviour makes clear that τ_D of `unbatch` and similar components must always be large enough so that the following components and loops are ready, before the next entity arrives. Taking a larger value does no harm, but by adding up delays or just trial and error one finds, that using 6ϵ works here, but 5ϵ doesn't. Additionally, only the `unbatch` component of the second furnace has to be adapted, while a default value of 2ϵ for the first `unbatch` works perfectly.

6 Conclusions

The central point of this investigation is the question, whether one can use NSA-DEVS without tinkering with a huge number of additional parameter adjustments of τ for the input delays and τ_D for the delays of transitory states. The previous discussion has shown that a consistent introduction of default values is crucial here: While setting $\tau = \epsilon$ and $\tau_D = \epsilon$ seems to work in many cases, components that emit trains of output values with infinitesimal time distances, such as a queue, batch and combine atomics, need special consideration. Often a value of $\tau_D = 2\epsilon$ is working and should be predefined in corresponding library components.

In special situations one has to enlarge these delays, but concrete values depend on the model details. Using a larger delay as general default could reduce the number of cases, where the user has to adapt it, but this apparent simplification is delusive.

The complex application studied here contains a total of 403 delay parameters, of which only one parameter had to be changed from its default value. This clearly supports the expectation that modeling with NSA-DEVS is feasible without drowning in a multitude of delay parameter adjustments.

In a current PhD project, the production chain described in this article is used as an application example, where it is integrated into the structure of an Experimental Frame [1]. The goal of the experiment is to optimize the structure and parameters of the model. As a first step, a parameter study has been carried out using the Design of Experiment method. This extended model works without further adaptations.

Considerations about the correct ordering of simultaneous events are not specific to NSA-DEVS, but are inherent to discrete-event modeling in general. Accordingly, other DEVS variants have different ways, how to cope with these situations. We see the advantage of NSA-DEVS in the clear-cut and versatile description of the ordering using infinitesimal time values. To make this work in practical applications, one needs support by the simulation tools.

The Matlab-based NSA-DEVS simulator used here supplies several debugging tools ranging from simple time stamps over debug output of individual atomic components up to complete output of internal simulator messages.

The analysis has shown opportunities to improve the implementation of some components. A prominent example is the smallestIn atomic that could be modified to output new values only, if the previous value changes. This reduces the number of output events preventing the infinite loop that has been encountered before. A more drastic change would be to include only one queue before the grinding machines and to distribute the parts after the queue. This would simplify the event structure and might be a reasonable idea for a production line design. But the whole point is to provide a tool that works for all modeling ideas, not to restrict the modeling to the tool capacities.

An open question from [6, 7] was, whether one needs port specific input delay times. The answer after this study is a definitive “no”: In all examples the default value of the input delay was sufficient for all input ports to get a reasonable model. A change of an input delay would be needed only to guarantee a certain order of unrelated input events. Besides, one could always resort to the workaround of inserting a simple delay component – e. g. a gain with factor 1 – before a specific input port.

This concludes the series of papers [4, 6, 7] that investigated the ideas behind NSA-DEVS as a foundation for component-based DEVS modeling. Building on RPDEVS [2], which made Mealy components simple and reliable, the NSA-DEVS approach was invented to add the robust modeling of causal event cascades. The definition of a simple abstract simulator, the careful analysis of standard examples and the implementation of a complex application have shown the soundness of its underlying ideas. The Matlab simulator and model base will be continually extended and provided freely from [11] to make NSA-DEVS-based modeling available for real-world applications.

References

- [1] Zeigler BP, Muzy A, Kofman E. *Theory of Modeling and Simulation*. San Diego: Academic Press, 3rd ed. 2019.
- [2] Preyser FJ, Heinzl B, Kastner W. RPDEVS: Revising the Parallel Discrete Event System Specification. In: *9th Vienna Int. Conf. Mathematical Modelling*. Wien. 2018; pp. 242–247.
- [3] Preyser FJ, Heinzl B, Kastner W. RPDEVS Abstract Simulator. *SNE Simulation Notes Europe*. 2019; 29(2):79–84. doi: 10.11128/sne.29.tn.10473.
- [4] Junglas P. NSA-DEVS: Combining Mealy Behaviour and Causality. *SNE Simulation Notes Europe*. 2021; 31(2):73–80. doi: 10.11128/sne.31.tn.10564.
- [5] Goldblatt R. *Lectures on the Hyperreals*. New York: Springer. 1998.
- [6] Jammer D, Junglas P, Pawletta T, Pawletta S. A Simulator for NSA-DEVS in Matlab. *SNE Simulation Notes Europe*. 2023;33(4):141–148. doi: 10.11128/sne.33.sw.10661.
- [7] Jammer D, Junglas P, Pawletta T, Pawletta S. Implementing Standard Examples with NSA-DEVS. *SNE Simulation Notes Europe*. 2022;32(4):195–202. doi: 10.11128/sne.32.tn.10623.
- [8] Larek R. Ressourceneffiziente Auslegung von fertigungstechnischen Prozessketten durch Simulation und numerische Optimierung (Resource-efficient Design of Manufacturing Process Chains by Simulation and Numerical Optimization). Ph.D. thesis, Universität Bremen. 2012.
- [9] Schmidt A. Variantenmanagement in der Modellbildung und Simulation unter Verwendung des SES/MB Frameworks (Variant Management in Modeling and Simulation using the SES/MB Framework). Ph.D. thesis, Hochschule Wismar / Universität Rostock. 2019.
- [10] Pawletta T, Schmidt A, Junglas P. A Multimodeling Approach for the Simulation of Energy Consumption in Manufacturing. *SNE Simulation Notes Europe*. 2017; 27(2):115–124. DOI: 10.11128/sne.27.tn.10377.
- [11] CEA Wismar. *NSA-DEVS on GitHub*. <https://github.com/cea-wismar/NSA-DEVSforMATLAB>.
- [12] Cellier FE, Kofman E. *Continuous System Simulation*. New York: Springer. 2006.
- [13] Schwatinski T, Pawletta T. A Quantization-based ODE Approximation and HPP-LGCA Approach to ARGESIM Benchmark C17 ‘SIR-type Epidemic’ in a DEVS Environment based on MATLAB. *SNE Simulation Notes Europe*. 2011;21(1):57–60. doi: 10.11128/sne.21.bn17.10053.
- [14] 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.
- [15] IEEE Design Automation Standards Committee. *Std 1164-1993, IEEE Standard Multivalued Logic System for VHDL Model Interoperability*. IEEE, New York, USA. 1993.

Fit for Duty Assessment of Driver Fatigue based on Statistical Modelling of Cardiovascular Parameters

Ciara Pircher¹, Martin Bachler^{1,2*}, Christer Ahlström³, Christopher C. Mayer²,
Bernhard Hametner²

¹Institute of Analysis and Scientific Computing, TU Wien, Wiedner Hauptstraße 8–10, 1040 Vienna, Austria

²Center for Health & Bioresources, AIT Austrian Institute of Technology, Giefinggasse 4, 1210 Vienna, Austria

*martin.bachler@ait.ac.at

³Swedish National Road and Transport Institute (VTI), Linköping, Sweden

SNE 33(4), 2023, 157-166, DOI: 10.11128/sne.33.tn.10663
Received: 2023-10-15; Revised: 2023-11-22
Accepted: 2023-11-24
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. Driver fatigue is a risk factor for road crashes. Fit for duty technologies could play a pivotal role in countering these crashes. Heart rate variability (HRV) and the pulse wave shape are influenced by the autonomic nervous system and are therefore affected by fatigue. This work focusses on modelling their relationship with fatigue and is based on data recorded in a simulated driving study. Six different multivariate linear regression models, using either stepwise variable selection or principal component analysis, are presented in this study. To account for differences in physiology, individual participant baselines for HRV and pulse wave parameters are introduced. Stepwise regression using any kind of baseline yields the most promising results. The most promising predictors are the ratio $\frac{LF}{HF}$ between low and high frequency components of HRV and heart rate. Finally, a stepwise regression model with a baseline, which has an adjusted R^2 statistic of 0.17, is proposed for further use. Nevertheless, further research with an extended dataset is necessary, incorporating a more diverse participant group and a higher number of recordings from severely sleepy drivers.

Introduction

Around 7% of European road crashes and around 13% crash-related injuries can be linked to driver fatigue [1]. A fit-for-duty assessment system, which can alert a driver of possible fatigue, has the potential to reduce the number of crashes, injuries and deaths on Europe's roads. Fit for duty assessments are typically based on ocular parameters [15, 16] or cognitive performance [17], and often evaluate the driver response to some sort of stimulus [18].

This work aims to develop a predictive model to estimate fatigue from cardiovascular parameters, derived from heart rate variability (HRV) and pulse wave shape. The model is meant for use in the field of commercial driving within the EU-funded PANACEA project, which stands for “practical and effective tools to monitor and assess commercial drivers’ fitness to drive”, and aims to take various driving impairments, such as alcohol or stress, into account.

1 Physiological Background

1.1 Heart Rate Variability

The autonomous nervous system (ANS), which keeps the body in homeostasis, a state of stable physical conditions, is constantly monitoring and correcting the heart rate (HR) to the needed pace via the sinus node. This means certain fluctuations in the time between two successive heartbeats are in fact healthy.

This variance indicates that our bodies can quickly adapt to environmental change or stressors and shows a degree of resilience. This fluctuation in time between successive heartbeats is termed the heart rate variability (HRV). [3]

Parameters derived from HRV can be extracted from Electrocardiography (ECG) and are usually separated in time and frequency domain parameters. An important measure for most time-domain HRV parameters is the normal-to-normal interval (NNI), which is a time series describing the time differences between successive normal heartbeats. After transforming NNI to the frequency domain, the power in certain frequency bands are widely used parameters. A description of used time and frequency domain parameters is given in table 1.

Table 1: Overview of time (1-4) and frequency (5-10) domain HRV parameters derived from ECG data. Units are given in parenthesis. [14]

	HRV Parameter	Description
1	<i>mean HR</i> (bpm)	Mean heart rate (HR) throughout a recording
2	<i>SDNN</i> (ms)	Standard deviation of NNIs (i.e. the square root of variance of NNIs)
3	<i>RMSSD</i> (ms)	Root mean square of successive differences of NNIs
4	<i>pNN50</i> (%)	Percentage of successive NNIs, that differ by more than 50 ms
5	<i>TP</i> (ms ²)	Total power in all frequency bands
6	<i>LF</i> (ms ²)	Power in the low frequency band (0.04 – 0.15 Hz)
7	<i>LFnorm</i> (-)	LF power divided by absolute power of LF+HF
8	<i>HF</i> (ms ²)	Power in the high frequency band (0.15 – 0.4 Hz)
9	<i>HFnorm</i> (-)	HF power divided by absolute power of LF+HF
10	<i>LF/HF ratio</i> (-)	Ratio of low frequency and high frequency power

As the body prepares for sleep, the heart rate decreases, allowing for more variability between beats, and the parasympathetic branch of the ANS becomes dominant while activity in the sympathetic branch of the ANS decreases [4]. Thus, we hypothesize that as fatigue arises, SDNN and HF should increase due to higher parasympathetic activity, whereas LF, heart rate and the $\frac{LF}{HF}$ ratio, which is said to describe the balance between the branches of the ANS, should decrease due to lower sympathetic activity [5].

Even though HRV parameters, especially $\frac{LF}{HF}$ ratio, are appealing parameters for fatigue assessment due to their physiological interpretation, they tend to show some controversy. There are inconsistencies in findings for all HRV parameters in connection to fatigue [19]. Concerning the frequency-domain HRV parameters, it should be noted that multiple different procedures are used to estimate the power spectrum and the applied method is often not clarified. Due to various anatomical factors, such as age or sex, there can also be large differences between individuals in HRV parameters [9, 10].

1.2 Pulse Wave

The ejection of blood from the heart causes a pressure wave that is partly reflected as it propagates through the arterial system. The pulse wave is the superposition of this pressure pulse and its reflections. Pulse arrival time (PAT) is the time from a point in the ECG, usually the prominent R-peak, to the detection of the pulse wave in a certain location of the body, in this case the finger. The pulse wave can be measured using photoplethysmography (PPG). [11]

Even though pulse waves are dependent on measurement location and individual factors, they mostly have similar main features that can be extracted as parameters. A general depiction of the pulse wave is shown in figure 1. Characteristic points of the pulse wave include the onset (P_O , the point before blood pressure begins to rise), diastolic blood pressure (P_{dia} , the minimal blood pressure), systolic blood pressure (P_{sys} , the first peak of blood pressure), the dicrotic wave amplitude (P_{dwa} , the second peak of the wave) and the dicrotic notch (P_{notch} , the trough between first and second peak of the wave) [26]. The total pulse duration (TPD) t_T is measured as the time from wave onset to the onset of the next wave. Descriptions of the parameters derived from the pulse wave are given in table 2.

It has been shown, that sleep deprivation affects blood pressure and therefore also the pulse wave [20]. Nevertheless, only one study was found that links changes in the pulse wave shape parameters to fatigue. An evaluation of sleepiness while flying, rather than driving, showed a significant increase in PAT, systolic time and diastolic time, which is the time from wave onset P_O to the diastolic peak P_{dwa} [12].

Table 2: Overview of pulse wave parameters derived from PPG and ECG data. [25, 27]

Pulse Wave Parameter	Description
t_T	Total pulse duration (TPD): time from wave onset P_O to the onset of the next wave
t_{sys}	Time from wave onset to systolic pressure P_{sys}
t_{sys_rel}	Time from wave onset to systolic pressure P_{sys} , relative to the TPD t_T
t_{notch}	Time from wave onset to dirotic notch P_{notch}
t_{notch_rel}	Time from wave onset to dirotic notch P_{notch} , relative to the TPD t_T
t_{dwa_rel}	Time from wave onset to dirotic wave amplitude P_{dwa} , relative to the TPD t_T
P_{dwa_sys}	Dirotic wave amplitude relative to systolic blood pressure: $\frac{P_{dwa}}{P_{sys}}$
P_{notch_sys}	Amplitude of the dirotic notch relative to systolic blood pressure: $\frac{P_{notch}}{P_{sys}}$
P_{notch_dwa}	Amplitude of the dirotic notch relative to dirotic wave amplitude: $\frac{P_{notch}}{P_{dwa}}$
PAT	Pulse arrival time

2 Methods

2.1 Data Collection and Processing

The data collection was conducted by the Swedish National Road and Transport Research Institute (VTI) in Linköping, Sweden. In total, 30 male professional drivers, who did not work nights and who are free of motion sickness and sleep disorders, completed six driving simulation tasks each. While the primary focus of the pilot trial was to examine effects of social drinking in the evening (target blood alcohol content (BAC) of 0.5‰) on next-day driving performance, the secondary focus was on fatigue data and modelling.

Driving tasks took approximately 35 minutes and were completed in a driving simulator in three different conditions: a control condition (C), where participants were under no known influence, a condition for the effect of alcohol (condition A), where drivers were intoxicated for half the measurements, and measurements conducted the day after drinking (condition B). Details concerning the measurement conditions are shown in table 3. Figure 2 shows the driving simulator and examples of scenery shown during the driving tasks.

Fatigue is measured using the subjective nine-point Karolinska Sleepiness Scale (KSS), depicted in table 4.

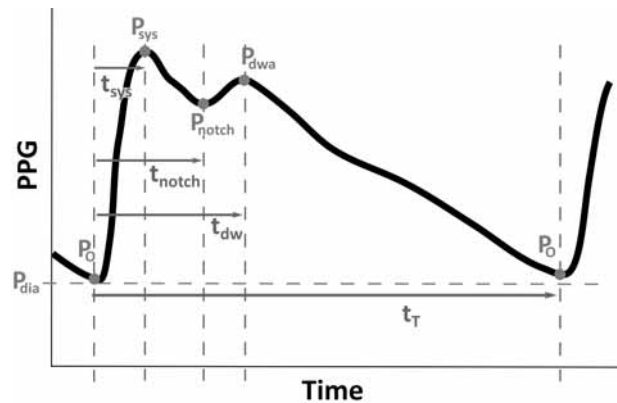


Figure 1: The image shows all characteristic points of the pulse wave (labelled in green) and all significant time durations (labelled in blue) used in this study.

Table 3: Summary of basic parameters for the different conditions in which each participant completed the simulated driving exercises. The blood alcohol content is abbreviated by BAC.

Condition	Purpose	Influence	Time of Day
A	Alcohol	BAC 0.3‰- 0.7‰ for half the drives	3 p.m. - 9.30 p.m
B	Day After	Residual Alcohol	7 a.m. - 1 p.m
C	Control	None	7 a.m. - 1 p.m



Figure 2: Left: The driving simulator used in the data collection. Right: Two examples of the simulated driving environment in rural (top) and urban (bottom) surroundings.

Drivers were asked to rate their level of fatigue on the KSS before and after driving. Using an objective ground truth of sleepiness would have been favourable, but available objective technologies suffer from intra- and interindividual differences and small effect sizes, whereas KSS has been found to be the measure of driver sleepiness least affected by inter-individual variations [2].

AIT Austrian Institute of Technology’s proprietary device, the SmartPWA, was used to record ECG and PPG signals both before and after driving. Measurements while driving are not possible, since the device must be held with both hands for recording signals. A detailed description of the device can be found in Mengden et al. [13]. In accordance with the standards of measurement for heart rate variability [14], at least two minutes of ECG and PPG data were recorded for each measurement in this trial.

For HRV measures in the frequency domain, the series of NNIs is transformed using the Lomb-Scargle-periodogram without interpolation [22, 23]. From this periodogram, the power in the low frequency band (0.04 - 0.15 Hz), the power in the high frequency band (0.15 - 0.04 Hz) as well as the total power are derived using the integral in the respective intervals. Data processing and modelling were conducted in MATLAB R2022b (The MathWorks Inc., Natick, USA).

Table 4: Levels of the Karolinska Sleepiness Scale (KSS) [2]

Level	Description
1	Extremely alert
2	Very alert
3	Alert
4	Rather alert
5	Neither alert nor sleepy
6	Some signs of sleepiness
7	Sleepy, but no effort to keep awake
8	Sleepy, some effort to keep awake
9	Very sleepy, great effort to keep awake, fighting sleep

2.2 Modelling

In addition to the ECG and PPG parameters already described, age, height, and weight were also included as predictors in the regression models. The number of parameters is too large to sensibly include all in one predictive model, which raises the question, which

parameters attribute most to accurate prediction of fatigue. MATLAB’s predefined functions for dimension reduction using principal component analysis (PCA) and stepwise variable selection, respectively, were used to determine the most valuable predictors and generate multivariate linear regression models.

Since alcohol is a known confounder of HRV [21], all models using no baseline were trained on data from condition C and, due to lack of more uninfluenced data, tested on data from condition A and B.

The individual differences in HRV and pulse wave parameters used as predictors in the generated regression models can have a huge effect on the generality of these models. Therefore, two different versions of an individual baseline were pursued: a fixed baseline (F) and a dynamic baseline (D).

Fixed Baseline (F). For each individual, the first measurement taken in control condition C, before driving, is used as the baseline. The training data then consists of the differences between any other measurement and the allocated participant baseline.

However, using recordings from condition C as a baseline, does not leave enough condition C recordings to train a model. Instead, for each participant, the two measurements for a given recording time (before or after driving) and a given condition (A,B or C) are randomly divided between the test and training data set.

Dynamic Baseline (D). The measurement before driving serves as a baseline in values for each participant for this particular drive. This baseline is dynamic in the sense that for each driving simulation a new participant baseline is set. The model is trained on the differences between before and after driving for data measured in condition C. Hence, the focus lies on the change in parameters throughout a simulated drive. The model is tested on the differences of parameters between before and after driving for data measured in conditions A and B.

Modelling Approaches. All generated models use combinations of the HRV and pulse wave parameters, presented in tables 1 and 2, as well as metadata (height, weight or age) to predict the level of fatigue on the KSS.

Six different approaches, depicted in figure 3, were pursued when generating multivariate linear regression models.

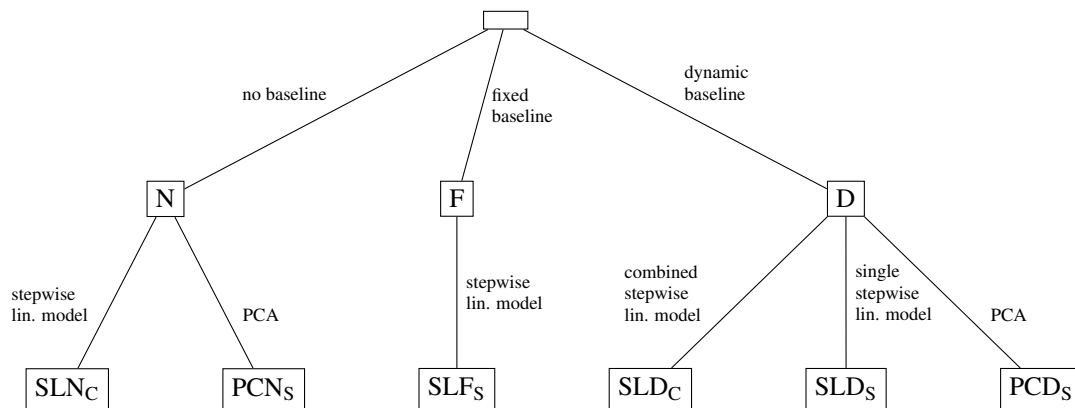


Figure 3: The tree represents an overview of different choices of baselines and modelling approaches for generating a model. A model can use no baseline (N), a fixed baseline (F) or a dynamic baseline (D). Applied methods can be principal component analysis (PC-) or stepwise linear regression (SL-). The subscript C indicates that the model is a combination of two models, where HRV and pulse wave data were modelled separately, while S indicates that one single model was generated from all data.

Each approach consists of a choice of baseline (fixed (-F), dynamic (-D) or none (-N)) and a choice of modelling method (PCA (PC-) or stepwise variable selection (SL-)). The models are given a three-letter name, where the first two indicate the chosen method while the last indicates the choice of baseline.

In some cases, HRV and pulse wave data are modelled separately, since, due to lack of signal quality, there are many missing values in pulse wave parameters. Generating separate models for the parameter groups allows the use of a larger training set for the HRV component of the model.

The final prediction for such models, which are in fact a combination of two models, is set as the average of both contributing predictions.

Models that are actually a combination of two separate models for HRV and pulse wave data are marked with a subscript C, while those generated as a single model from all data simultaneously are marked with a subscript S.

Evaluation. The models are evaluated using the F -test, which determines the statistical significance of the relationship between a group of predictors and the response. The relationship given by a model is significant, if the p -value determined by the F -test is below the level of significance $\alpha = 0.05$. Residual plots are used to detect systematic error or non-normality of errors and residuals are tested for normality using the Anderson-Darling test.

Models are also compared to each other with respect to quality of fit, using the adjusted R^2 statistic as well as root mean square error (RMSE), based on the difference between measured and predicted KSS, on test and training data as main indicators of goodness of fit.

3 Results

The median age of the drivers was 40 years with an interquartile range (IQR) of 12 years. The participants had a median height of 183 cm with an IQR of 9 cm. The median weight of drivers was 91 kg with an IQR of 21 kg.

Table 5 shows a summary of model results. The results of the F -test indicate that all but one model are statistically significant. The model PCN_S is not statistically significant. While the model SLD_S has an adjusted R^2 statistic of 0.6, all other models are below 0.25. The root mean square error of the models ranges from 0.76 to 1.26 for training data. For test data RMSE is between 1.3 and 1.75.

The tables 6 and 7 show the RMSE for data from each of the conditions separately. In general, the prediction error seems to be highest in condition B, the day after drinking, while it is lowest on data from condition C, on which most models were trained.

Graphical residual analysis using residual plots did not reveal any inappropriate model choices or correlated errors.

Table 5: The table shows key characteristics of quality of fit, the adjusted R^2 statistic and the root mean square error (RMSE) on both training and test data on the KSS scale (1-9), for each of the models discussed. Additionally, the p -value of the F -test versus a constant model is given for each of the models.

Model	Adjusted R^2	Training RMSE	Test RMSE	F -Test
SLN _C	0.21	1.11	1.55	$p_{HRV} = 0.0053$ $p_{PW} = 0.0274$
SLD _C	0.17	1.19	1.41	$p_{HRV} = 0.0035$ $p_{PW} = 0.0143$
SLD _S	0.60	0.76	1.75	$p = 0.0002$
SLF _S	0.14	1.14	1.3	$p = 0.0037$
PCN _S	0.07	1.26	1.55	$p = 0.4924$
PCD _S	0.25	1.19	1.52	$p = 0.0235$

Table 6: RMSE for KSS estimates on the scale of 1 to 9 of SLN_C, SLF_S and PCN_S models. The RMSE is given for each data category separately. The training data set of SLN_C and PCN_S is marked by an asterisk.

Data	SLN _C	SLF _S	PCN _S
C (before)*	1.04	-	1.26
C (after)	1.21	1.08	1.23
B (before)	1.68	1.35	1.49
B (after)	1.78	1.52	1.65
A (before)	1.35	1.03	1.35
A (after)	1.61	1.08	1.69

Table 7: RMSE for KSS estimates on the scale of 1 to 9 of models using dynamic baselines, i.e. SLD and PCD_S models. The RMSE is given for each data category separately. The training data set is marked by an asterisk.

Data	SLD _C	SLD _S	PCD _S
C*	1.19	0.76	1.19
B	1.52	1.48	1.68
A	1.30	1.97	1.36

While the residuals of the models SLN_C, SLD_C, SLD_S and the principal component models PCN_S and PCD_S passed the Anderson-Darling test of normality, those of the model SLF_S did not.

Table 8 gives an overview of the variables included in each of the generated stepwise linear models as well as their estimated coefficient values.

The most important variables in the generated regression models seem to be age and heart rate, which are both selected in three models (with a statistical significant relationship in two of them), as well as the $\frac{LF}{HF}$ ratio for HRV data, which is selected in all models and is significant in two of these. Concerning pulse wave parameters, systolic time and total pulse duration are chosen with a statistically significant coefficient comparatively often: in two and three models, respectively.

Table 8: This table shows an overview of the variables chosen by each stepwise regression model as well as their computed coefficients. An asterisk indicates statistical significance at the level $\alpha = 0.05$ in the corresponding model. Fields of coefficients, that were not selected, are shaded in dark grey, while those that were not statistically significant are shaded in light grey.

Variable	SLN _C	SLD _C	SLD _S	SLF _S
intercept	-1.5800	0.3155	-7.2307	4.8773
age	-0.0915*		0.0379	-0.0378*
height	-0.0413		0.0453	0.318
weight			-0.0298	
mean HR	-0.0688*		-0.0966*	-0.0736
LF	418.15		-1175.50*	-394.46
HF			5567.10*	
$\frac{LF}{HF}$ ratio	0.1858	0.2751*	1.0477*	-0.1314
RMSSD				
SDNN	-0.0967*			
pNN50	-2.282		-13.4670*	
TP	358.86			325.96*
LFnorm				4.3528*
HFnorm				
t_T	0.0206*			-0.0093*
t_{notch}				
t_{sys_rel}	65.2190*			
t_{notch_rel}			22.762*	
t_{dwa_rel}		8.4928		
t_{sys}	-0.0690*	-0.0160*	-0.0237*	
P_{dwa_sys}	29.6110	5.3830	6.3948*	
P_{notch_sys}	-32.1640			
P_{notch_dwa}	18.7490			
PAT	-0.0142	0.0598*		

The principal component model with no baseline (PCN_S) uses eight of the 23 computed principal components, none of which are statistically significant.

The model PCD_S uses five principal components, two of which have a significant relationship to the response.

The model SLD_C is described in greater detail, since this is later chosen as the most favourable model. Predictions are calculated using the formula

$$KSS_{est} = \frac{(KSS_{HRV} + KSS_{PW})}{2}$$

where the KSS values from HRV and PW data are estimated by

$$KSS_{HRV} = K_{HRV} + c_1 \frac{LF}{HF}$$

$$KSS_{PW} = K_{PW} + c_2 t_{dwp_rel} + c_3 t_{sys} + c_4 P_{dwp_sys} + c_5 PAT.$$

K_{HRV} and K_{PW} denote the estimated intercept included in the corresponding models while the coefficients are referred to as c_j , $j = 1, \dots, 5$. Figure 4 shows the estimated KSS values, fitted by the model SLD_C , plotted against the corresponding measured KSS values.

4 Discussion

This work aims to predict fatigue on the KSS from HRV and pulse wave parameters. Four different stepwise linear regression models, the coefficients of which can be found in table 8, and two models using principal component analysis were generated. Using linear regression resulted in fatigue predictions that were on average about 1.5 KSS units wrong. Principal component analysis and regression did not lead to improvement compared to stepwise linear regression models, especially with respect to statistical significance. This could be due to the complexity of the cardiovascular system and the multitude of confounding factors, many of which could not be included in the regression data. Due to the large individual differences in HRV and pulse wave data, using a baseline improved model results.

While many studies investigate the connection between single HRV measures and fatigue [4, 5, 8, 19], only one study was found, that evaluates this relationship for pulse wave shape parameters [12]. In these previous studies HRV and pulse wave shape parameters are used for continuous fatigue monitoring during driving. As this study attempts to assess driver sleepiness prior to driving and no previous research on predictive regres-

sion models for fatigue based on cardiovascular parameters was found, the results of this study are difficult to put into context.

The variables age, heart rate and $\frac{LF}{HF}$ -ratio are often included as variables, while also being statistically significant in the stepwise regression models of this work. They are therefore considered to be the most important HRV parameters connected to fatigue. Similarly, when looking at the results of this study, systolic time and total pulse duration are very important pulse wave parameters for fatigue assessment.

These parameters are also considered to be important variables in previous studies, however, for the systolic time and frequency-domain HRV measures, the coefficient signs are mostly the opposite of what could have been expected from previous research [6, 8, 12]. In comparison to literature, the influence of height and weight is smaller than expected [10]. They are not chosen with a statistically significant coefficient in any model. RMSSD is not included at all and therefore seems to be of low importance. PAT is chosen with a statistically significant coefficient in one model, but seems to be less sensitive to changes in fatigue than expected.

Both, previous studies and the stepwise regression models of this study suggest that the $\frac{LF}{HF}$ ratio is of utmost importance for fatigue prediction, but contradictory results pose challenges in its use as a predictor. While most studies, such as [6], show a negative trend for rising sleepiness, some research, such as Rodriguez-Ibañez et al. [7], shows the opposite or, as in Abtahi et al. [8], finds no significant change. In this work the $\frac{LF}{HF}$ ratio is included significantly with a positive trend in two models, SLD_S and SLD_C . Contradictory results could be caused by confounding factors or, as suggested in the standards of measurement for HRV [14], could be the result of varying methods in use to obtain the frequency-domain measures. Since most studies do not clarify the applied method, the exact influence can not be determined. Alternatively, increased stress due to fighting fatigue, when driving on real roads compared to a simulated environment, could increase sympathetic activity and therefore affect HRV and limit the comparability of studies [8].

For the model SLD_S the heart rate and HF power are included as expected [8]. The models SLF_S , SLN_C and SLD_C each include one variable as the literature review would suggest (TP, HR and PAT, respectively) [8, 12].

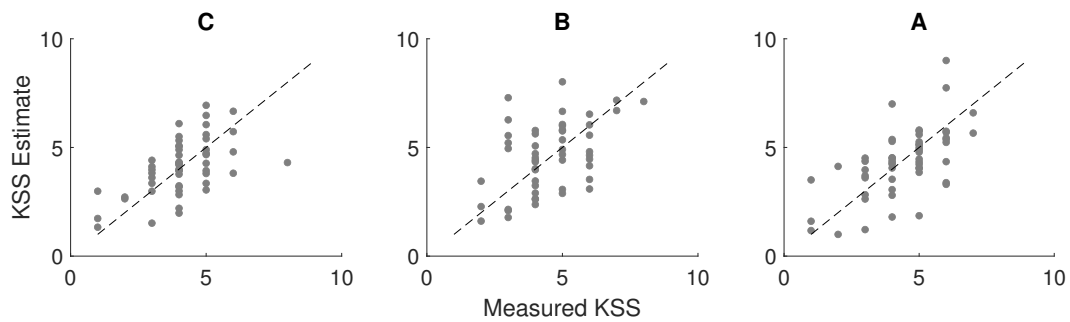


Figure 4: KSS values fitted by the model SLD_C are plotted against the corresponding measured KSS values in each group. The black dotted line indicates the line of equality between measures and estimates.

Even though all residual plots look acceptable, the residuals of the model SLF_S failed the test of normality, indicating possible systematic error. Therefore, its results should be used with caution. The model SLD_S on the other hand, shows signs of overfitting. The adjusted R^2 statistic indicates a higher portion of explained variance than could be expected in such a complex system and especially the high difference between test and training RMSE causes doubts, whether this is a suitable choice.

The model SLF_S has an acceptable R^2 value and retains a higher generality than other models presented in this study. However, the better quality of fit of SLF_S must be seen in the context of test and training data. While all other models were trained on condition C and tested on conditions A and B, both training and test data sets of the SLF_S model contained measurements from all conditions.

In the context of predictions, low p -values and RMSE are essential. Therefore, even though SLN_C has low training error and high adjusted R^2 , it may not be suited for the intended use, since only 6 out of 15 variable coefficients are statistically significant. The combination of all variables is considered to be significant at the level $\alpha = 0.05$, but the p -value is higher than that achieved by other models.

The model SLD_C seems to strike a balance, where a good amount of variance is explained through a small number of variables, while statistical significance, prediction error and the normality of residuals are all acceptable. Nevertheless, the results should be interpreted with caution, since this model includes some variables, most notably the $\frac{LF}{HF}$ ratio and systolic time, in a different manner, i.e. opposite sign of the coefficient, than the majority of previous research.

One limitation of this work is that the data used for the purpose of generating the regression models is not perfectly suited to the task. Considering the fact that predicting high KSS values is of most interest in the context of driving, it is unfortunate, that over 90% of all recorded KSS values are below 7.

During the entire trial, no participant was tired enough to evaluate themselves at the highest KSS value of 9. This fact does not allow to generate or even test a model, that predicts fatigue accurately at the top end of the scale, thus a generally valid model.

Additionally, the training data set is rather small, after removing data influenced by alcohol. Therefore, no data without known influences can be reserved for model testing, which makes the interpretation of results difficult. Multiple factors, such as age, sex, shift work or certain medical conditions can affect HRV and pulse wave parameters and should also be accounted for [14, 24].

5 Conclusion

In the context of quantifying the relationship between fatigue and physiological parameters of the cardiovascular system, which are sensitive to changes in fatigue due to their connection to the autonomic nervous system, the presented linear regression models using stepwise variable selection produce promising results.

Even with the restriction of a small data set with avoidable confounding factors, this work shows that the prediction of fatigue on the KSS scale through a regression model using cardiovascular parameters is not only feasible in theory, but also in practice.

Of course, the methods and models presented and discussed in this work need to be refined, especially by incorporating larger and more heterogeneous data sets, before fatigue assessment for commercial drivers can be used at a large scale. The dataset included only healthy, male participants, mainly between the ages of 30 and 50. Even though the homogeneous participant group has the advantage that much variability can be avoided at such an early stage, it also means the models must be generalised and re-evaluated to be applicable to the entire adult population.

Acknowledgement

This work was carried out in the PANACEA project which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 953426. We are also very grateful to the data collection team at VTI.

Ciara Pircher was supported through a scholarship within the Talents programme of the Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology.

References

- [1] Gonçalves M, Amici R, Lucas R, et al. Sleepiness at the wheel across Europe: a survey of 19 countries. *Journal of Sleep Research*. 2015; 24(3): 242–253. doi: 10.1111/jsr.12267
- [2] Åkerstedt T, Anund A, Axelsson J, Kecklund G. Subjective sleepiness is a sensitive indicator of insufficient sleep and impaired waking function. *Journal of Sleep Research*. 2014; 23(3): 240–252. doi: 10.1111/jsr.12158
- [3] Shaffer F, Ginsberg J P. An overview of heart rate variability metrics and norms. *Frontiers in public health*. 2017; 5(258). doi: 10.3389/fpubh.2017.00258
- [4] Persson A, Jonasson H, Fredriksson I, Wiklund U, Ahlström C. Heart Rate Variability for Classification of Alert Versus Sleep Deprived Drivers in Real Road Driving Conditions. *IEEE Transactions on Intelligent Transportation Systems*. 2021; 22(6): 3316–3325. doi: 10.1109/TITS.2020.2981941
- [5] Burlacu A, Brinza C, Brezulanu A, Covic A. Accurate and early detection of sleepiness, fatigue and stress levels in drivers through Heart Rate Variability parameters: a systematic review. *Reviews in cardiovascular medicine*. 2021; 22(3): 845–852. doi: 10.31083/j.rcm2203090
- [6] Awais M, Badruddin N, Drieberg M. A non-invasive approach to detect drowsiness in a monotonous driving environment. *TENCON 2014-2014 IEEE Region 10 Conference*. 2014. doi: 10.1109/TENCONSpring.2014.6863035
- [7] Rodriguez-Ibañez N, García-Gonzalez M A, de la Cruz M A F, Fernández-Chimeno M, Ramos-Castro J. Changes in heart rate variability indexes due to drowsiness in professional drivers measured in a real environment. *2012 Computing in Cardiology*. 2012; 913–916.
- [8] Abtahi F, Anund A, Fors C, Seoane F, Lindecrantz K. Association of drivers' sleepiness with heart rate variability: A pilot study with drivers on real roads. *EMBECE & NBC 2017*. 2017; IFMBE Proceedings, vol 65. doi: 10.1007/978-981-10-5122-7_38
- [9] Lutfi M F, Sukkar M Y. The effect of gender on heart rate variability in asthmatic and normal healthy adults. *International journal of health sciences*. 2011; 5(2): 146–154.
- [10] Lutfi M F, Sukkar M Y. Relationship of height, weight and body mass index to heart rate variability. *Sudan Med J*. 2011; 47(1): 14–19.
- [11] Tamura T, Chen W. Blood Pressure. *Seamless Healthcare Monitoring*. Springer; 2018. p 103–127.
- [12] Majumder S, Verma A K, Wang C et al. Using photoplethysmography based features as indicators of drowsiness: Preliminary results. *2019 Design of Medical Devices Conference*. 2019. American Society of Mechanical Engineers. doi: 10.1115/DMD2019-3236
- [13] Mengden T, Bachler M, Sehnert W, Marschall P, Wassertheurer S. Device-guided slow breathing with direct biofeedback of pulse wave velocity—acute effects on pulse arrival time and self-measured blood pressure. *Blood Pressure Monitoring*. 2023; 28(1): 52–58. doi: 10.1097/MBP.0000000000000628
- [14] Malik M. Heart rate variability: Standards of measurement, physiological interpretation, and clinical use: Task force of the European Society of Cardiology and the North American Society for Pacing and Electrophysiology. *Annals of Noninvasive Electrocardiology*. 1996; 1(2): 151–181. doi: 10.1111/j.1542-474X.1996.tb00275.x
- [15] Ahlström C, Nyström M, Holmqvist K, Fors C, Anund A, Kecklund G, Åkerstedt T. Fit-for-duty test for estimation of drivers' sleepiness level: Eye movements improve the sleep/wake predictor. *Transportation Research Part C: Emerging Technologies*. 2013; 26: 20–32. doi: 10.1016/j.trc.2012.07.008

- [16] Di Stasi L, Renner R, Catena A, Cañas J, Velichkovsky M, Pannasch S. Towards a driver fatigue test based on the saccadic main sequence: A partial validation by subjective report data. *Transportation Research Part C: Emerging Technologies*. 2012; 21(1): 122–133. doi: 10.1016/j.trc.2011.07.002
- [17] Perelli L. CogSpeed: A Proposed Technology for Instantly Measuring Cognitive Performance in a Laboratory or a Real World Setting. 2023.
- [18] Cori J, Manousakis J, Koppel S, Ferguson S, Sargent C, Howard M, Anderson C. An evaluation and comparison of commercial driver sleepiness detection technology: a rapid review. *Physiological Measurement*. 2021; 42(7). doi: 10.1088/1361-6579/abfbb8
- [19] Lu K, Dahlman A, Karlsson J, Candefjord S. Detecting driver fatigue using heart rate variability: A systematic review. *Accident Analysis & Prevention*. 2022; 178. doi: 10.1016/j.aap.2022.106830
- [20] Kato M, Phillips B, Sigurdsson G, Narkiewicz K, Pesek C, Somers V. Effects of Sleep Deprivation on Neural Circulatory Control. *Hypertension*. 2000; 35(5): 1173-1175. doi: 10.1161/01.HYP.35.5.1173
- [21] Romanowicz M, Schmidt J, Bostwick J, Mrazek D, Karpyak V. Changes in heart rate variability associated with acute alcohol consumption: current knowledge and implications for practice and research. *Alcoholism: Clinical and Experimental Research*. 2011; 35(6): 1092-1105. doi: 10.1111/j.1530-0277.2011.01442.x
- [22] Bachler M, Sehnert W, Mikisek I, Wassertheurer S, Mengden T. Non-invasive quantification of the effect of device-guided slow breathing with direct feedback to the patient to reduce blood pressure. *Physiological Measurement*. 2020; 41(10): 1092-1105. doi: 10.1088/1361-6579/abb320
- [23] Fonseca D, Netto A, Ferreira R, De Sa A. Lomb-scargle periodogram applied to heart rate variability approach. *ISSNP Biosignals and Biorobotics Conference: Biosignals and Robotics for Better and Safer Living (BRC)*. 2013.
- [24] Skornyakov E, Gaddameedhi S, Paech G, Sparrow A, Satterfield B, Shattuck N, Van Dongen H. Cardiac autonomic activity during simulated shift work. *Industrial health*, 57(1): 118-132. 2019.
- [25] Korpas D, Halek J, Doležal L. Parameters describing the pulse wave. *Physiological Research*, 58(4): 473-479. 2009.
- [26] Solà J, Delgado-Gonzalo R. *The Handbook of Cuffless Blood Pressure Monitoring*. Springer; 2019.
- [27] Tamura T, Chen W. *Seamless Healthcare Monitoring*. Springer; 2018.

Integrating Reinforcement Learning and Discrete Event Simulation Using the Concept of Experimental Frame: A Case Study With MATLAB/SimEvents

Thorsten Pawletta^{*}, Jan Bartelt

Research Group Computational Engineering and Automation (CEA), Wismar University of Applied Sciences, Philipp-Müller-Straße 14, D-23966 Wismar, Germany; ^{*}*thorsten.pawletta@hs-wismar.de, jan.bartelt@hs-wismar.de*

SNE 33(4), 2023, 167-174, DOI: 10.11128/sne.33.sw.10664
 Selected ASIM WS 2023 Postconf. Publication: 2023-10-15
 Received Revised Improved: 2023-11-22; Accepted: 2023-11-25
 SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
 ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. Reinforcement Learning (RL) is an optimization method characterized by two interacting entities, the agent and the environment. The environment is a Markov Decision Process (MDP). The goal of RL is to learn how an agent should act to achieve a maximum cumulative reward in the long-term. In discrete-event simulation (DES), the dynamic behavior of a system is represented in a model (DESM) that is executed via a simulator. The concept of Experimental Frame (EF) provides a structural approach to separating the DESM into the Model Under Study (MUS) and its experimental context. Here, we explore the integration of a discrete event MUS as an environment for RL using the concept of EF. After discussing the methodological framework, a case study using MATLAB/Simulink and the SimEvents blockset is considered. The case study starts with an introduction of the discrete-event MUS for which a control strategy shall be developed. The MUS is reused in three experiments using specific EFs. First, an EF for the design of a heuristic control strategy with ordinary simulation runs is presented. Then, based on the methodological approach, specifics of the EF are considered when using a self-implemented Q-agent and the RL toolbox of MATLAB/Simulink.

Introduction

In modeling and simulation (M&S), a model describes the dynamic behaviour of a real or virtual system. The execution of the model is performed using a simulator. In the versatile use of a model, it should be developed independently from the context of use.

The reference to a concrete experiment can be mapped by an Experimental Frame (EF).

An EF specifies the conditions under which a system is observed or a model experimented with (Zeigler [12], Zeigler et al. [14], Traore and Muzy [11]). The model used is called the Model under Study (MUS). Depending on the EF, the same MUS can be used in different experimental contexts, such as a parameter study, sensitivity analysis, optimization, etc. The EF and MUS form the simulation model (SM). Discrete event simulation models (DESM) are characterized by a finite number of states over a continuous time base.

The EF implements the interface for a Simulation-Based Experiment (SBE). Inspired by Breitenecker's [1] approach to structuring SBEs, Pawletta et al. [5] and Schmidt [7] introduced the concept of Simulation Method (SimMeth) and Experiment Method (ExpMeth). The SimMeth controls the execution of the simulation runs via a simulator and ExpMeths are arbitrary numerical methods. ExpMeths are used for pre- and post-processing or to control the SimMeth, such as in simulation-based optimization experiments (Carson and Maria [2]; Schmidt [7]).

Reinforcement Learning (RL) (Sutton and Barto [8]) in combination with a dynamic system simulation can be considered as a SBE. However, RL is an optimization method for Markov Decision Processes (MDPs). The MDP is modeled as an environment and an agent acts as a controller. The goal is to learn how the agent should act to achieve a maximum cumulative reward in the long-term.

In contrast to a DESM, an MDP is a discrete time process and the time base is only used for the sequential ordering of states. Not all states of the MUS are usually of interest to the RL. Accordingly, the states of the MUS must be converted into MDP-compliant states.

Due to the methodological differences, the combination of the two methods, RL and discrete event simulation, often lead in practice to implementations that are difficult to maintain and MUS that are not generally usable.

Here, the practical integration of both methods using the concept of EF is explored by means of a case study and using MATLAB/Simulink as well as the SimEvents blockset (MathWorks [9]). We start with some basics to SBEs, EF, RL and the usage of RL in a SBE. Then, the MUS is introduced for which a control strategy is developed. To present the reusability of the MUS in the context of different experiments using specific EFs, we start with the design of a heuristic controller. This is followed by two experiments on RL-based controller design.

This work is based on Pawletta and Bartelt [6]. More details on the theoretical background and related work is provided there.

1 Basics

Based on Pawletta and Bartelt [6], we briefly review the basics of structuring SBEs, the RL method, and the use of RL as a method of an SBE.

1.1 Structuring Simulation-Based Experiments

Schmidt [7] divides SBEs into three classes. We consider only the first two classes. The execution of one or more simulation runs by a SimMeth constitutes a simple SBE, if the SimMeth is invoked directly by the user or a supervisory Experiment Control (EC). An EC defines the goals and steps of an experiment and automates the experiment execution.

In a complex SBE, the SimMeth is controlled by an ExpMeth, for example, by a numerical optimization method. Figure 1 shows the basic structure of a complex SBE. Both the SimMeth and ExpMeth define process parameters (P_{ExpM} , P_{SimM}).

The EF separates the MUS from a specific context of use to improve the reusability of the MUS. Formally, Zeigler [13] defines the function of an EF with the tuple.

$$EF = \langle T, I, C, O, \Omega_I, \Omega_C, SU \rangle \quad (1)$$

T represents the time base, I and O the set of input and output variables of the MUS (equivalent to I_{MUS} and O_{MUS} in Figure 1), C the set of run control variables, Ω_I the set of admissible input segments, Ω_C the set of admissible control segments, and SU the set of summary mappings. Set Ω_I refers to the input variables of the MUS and to the input/output relationships in the EF.

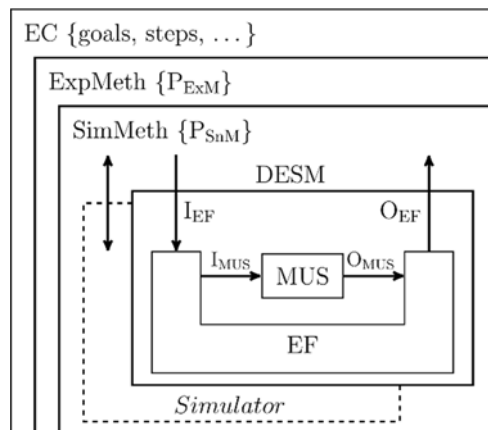


Figure 1. Basic structure of a complex SBE.

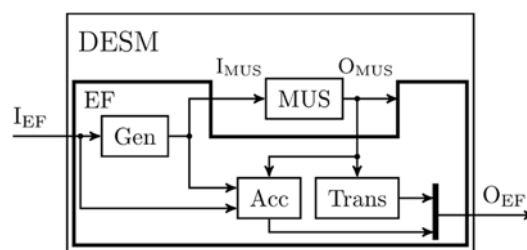


Figure 2. Basic structure of a DESM with MUS and EF.

An EF does not necessarily have to contain all three components and the coupling relationships are not fixed.

Set Ω_C defines the experimental constraints. The experiment objectives are mapped to *interest variables*. Set SU defines the determination of the interest variables based on the MUS outputs. The interest variables are typical output variables of the EF. The implementation of an EF is done using three types of components, as illustrated in Figure 2 (Zeigler [13]; Zeigler et al. [14]). The *generator* (Gen) initializes the configurable parameters of the MUS and calculates the input segments for the MUS which can also be inputs of the *Acceptor* (Acc) or *Transducer* (Trans). The Acc defines the admissible control segments and monitors their compliance. The output of the Acc is run control information. The Trans calculates the SU.

1.2 The Method of Reinforcement Learning

According to Sutton and Barto [8], RL focuses on the sequential decision-making by an agent that interacts with a real or virtual environment. The agent is trained by its interactions with the environment. The goal of RL is to learn a behavioral strategy $\pi: S \rightarrow A$ for the agent that assigns an action $a \in A$ to each state $s \in S$ of the environment.

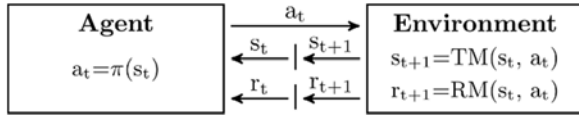


Figure 3. Basic RL framework.

Thus, the agent can act as a controller for the environment. Using RL, a distinction is made between the training and deployment of an agent, although the agent can continue learning during deployment. The basic RL framework is shown in Figure 3.

In model-free RL, the agent only knows the allowed action set A at the start of training. The states $s \in S$ of the environment are unknown to the agent. When an action $a_t \in A$ takes effect, the environment determines its next state s_{t+1} as well as a reward value r_{t+1} using a state transition model $TM: S \times A \rightarrow S$ and reward model $RM: S \times A \rightarrow R$. The next state and the reward value are sent back to the agent. The index t marks a sequence of states in the sense of a MDP. Through iterative interactions with the environment, the agent obtains information about possible states of the environment and the benefits of actions, gradually improving its behavioral strategy π . A variety of different learning strategies have been developed for RL agents such as Q-learning, Deep Q Networks etc.

We briefly consider Q-learning that uses formula (2) to learn a strategy π using a table function called the Q-matrix. A matrix element $Q(s, a)$ represents the estimated benefit of an action a_t when it is performed in the state s_t of the environment. The updated $Q(s_t, a_t)$ value of the current state/action tuple (s_t, a_t) is calculated from the previous $Q(s_t, a_t)$ value, the currently received reward r_{t+1} , and the maximum Q-value ($\max_a Q(s_{t+1}, a)$) of all possible actions in the currently received next state s_{t+1} . The variables α and γ are hyperparameters, i.e. they must be defined before training, but can still be adjusted during training.

$$Q(s_t a_t) \leftarrow Q(s_t a_t) + \alpha [r_{t+1} + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (2)$$

The training takes place in independent episodes. Each episode starts in an initial state s_0 of the environment and ends when a target state s_{target} or abort state s_{abort} is reached. At the beginning of the training, the agent selects an action $a \in A$ randomly. This is called *exploration*. As the learning process progresses, the agent increasingly uses the knowledge it has acquired to select an action which is called *exploitation*.

The ratio ε of exploration to exploitation is adjusted over the course of the training. After the completion of a defined number of episodes, the behavioral strategy $a = \pi(s)$ is derived from the training data.

1.3 Integrating Reinforcement Learning into a Simulation-Based Experiment

When integrating RL and dynamic system simulation, the MUS forms the environment for the RL agent. The goal of such an SBE is

- to learn the best possible behavioral strategy of the agent,
- to extract this strategy from the training data, and
- to use it as a controller for the MUS.

The first two items are defined with an ExpMeth *training* that controls a SimMeth to execute simulation runs. The ExpMeth training contains the following basic steps:

- Set the RL-specific experiment parameters PExM such as the learning rate, exploration rate, maximum number of episodes, Q-matrix etc.
- Set the simulation execution parameters PSnM for the SimMeth, such as the simulator to be used, the simulation time interval etc.
- Set the DESM parameters for the EF and the MUS and prepare the DESM for executing using a SimMeth.
- Initialize statistical variables, such as those used to record the total reward per episode etc,
- Compute the defined number of episodes, i.e. call the SimMeth into a loop to execute the DESM, update the statistical variables after each episode, and check whether to abort the training or continue with another episode.
- Determine and save the best policy π , and plot essential learning results.

Figure 4 shows the basic structure of a DESM with an EF for RL in the training phase. The variables τ and t represent the different time bases. τ is the continuous time of the MUS and t the discrete time for ordering the sequential states of the RL.

The input variables I_{EF} are initialized at the simulation start time τ_0 , at the beginning of an episode. Results are get back via O_{EF} at the *end of an episode (eoe)*.

The *Gen* is subdivided into three subcomponents. *Gen.G_{MUS}* initializes the parameters of the MUS at τ_0 and calculates input segments Ω_I for the MUS inputs $I_{MUS}(\tau)$ over the course of an episode.

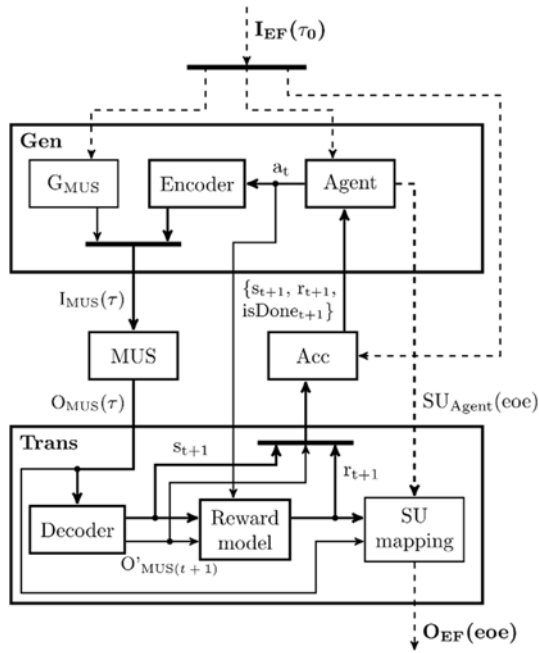


Figure 4. Basic structure of a DESM with a MUS and EF for RL in the training phase.

Gen.Agent maps the RL agent. It is part of the *Gen* because its output, an action a_t , produces inputs of the MUS. The agent's parameters are initialized via the input interface $I_{EF}(\tau_0)$. In addition to the ordinary agent inputs' new state s_{t+1} and the reward value r_{t+1} , the third input $isDone_{t+1}$ is a Boolean value that signals the end or cancellation of an episode. At the *eoe*, the agent creates a summary mapping SU_{Agent} that contains RL-specific values such as the *number of steps*, the *total reward*, or the strategy learned so far (e.g. the *Q-matrix*). The SU_{Agent} is passed to the *Trans*. *Gen.Encoder* defines a mapping $i(\tau)=h(a_t)$ to transform a single action value a_t into MUS compatible input values $i(\tau) \in I_{MUS}(\tau)$ as introduced by Choo et al. [3].

The *Trans* is also subdivided into three subcomponents. *Trans.Decoder* defines (i) the calculation of the *interest values* $O'_{MUS(t+1)}$ from the current outputs $O_{MUS}(\tau)$ of the MUS related to the time base of the RL (i.e. $O'_{MUS(t+1)}=f(O_{MUS}(\tau))$), and transformation of the interest values $O'_{MUS(t+1)}$ to a state s_{t+1} in the RL space (i.e. $s_{t+1}=g(O'_{MUS(t+1)})$). Thus, all interest values of the MUS are mapped into one state for the RL and for each particular interest value there is only one corresponding state in the RL space (Choo et al. [3]). The *Trans.Reward-model* maps the reward calculation. The reward value characterizes a state transition $s_t \rightarrow s_{t+1}$ in the RL space.

Defining the reward calculation is sometimes a difficult problem. Our own experiments showed that the reward value can sometimes be computed very efficiently based on the $O'_{MUS(t+1)}$ values. The component *Trans.SUmapping* implements the overall SU of an episode and passes it at the *eoe* to the output O_{EF} .

The *Acc* checks compliance with the constraints for the episode using defined run control information. Run control variables can be initialized via $I_{EF}(\tau_0)$. Typical run conditions to be monitored include the simulation interval $[\tau_0, \tau_{final}]$ of the MUS and thus the maximum duration of an episode, the detection of illegal states or the reaching of a target state. The *Acc* checks all the relevant quantities and sets the *isDone* value, before sending the tuple $(s_{t+1}, r_{t+1}, isDone_{t+1})$ to the *Gen.Agent*.

When deploying a learned strategy, we have to distinguish whether it is used with or without further learning of the agent. For an experiment *deployment without training* the EF simplifies as shown in Pawletta and Bartelt [6]. No explicit ExpMeth is required. The SimMeth is called directly in the EC according to the number of simulation runs to be executed.

2 Case Study

The basic implementation of the approach to integrate RL and discrete-event simulation introduced in Section 1.3 will be demonstrated by a case study using MATLAB/Simulink and the SimEvents blockset. The objective is to develop a control strategy for a MUS with discrete-event dynamics. First, the most general possible modeling of the MUS, i.e. without concrete references to an experiment, is discussed. Then, the same MUS is used in three experiments using different EFs: (i) to design a heuristic strategy, (ii) to learn a strategy with a self-implemented Q-agent, and (iii) to learn a strategy using MathWorks' dedicated RL toolbox (MathWorks [10]).

2.1 Model Under Study and General Objectives of the Control Design

The MUS is a simple server line consisting of an entity generator, a convertible operating unit, and two downstream servers connected in parallel with separate input queues as shown in Figure 5. The operating unit can process two types of entity ($jobType=1 / 2$). A separate processing time can be defined for each entity type (*procT1*, *procT2*). A retooling time (*retoolingT*) is necessary when the entity type is changed in the operation unit.

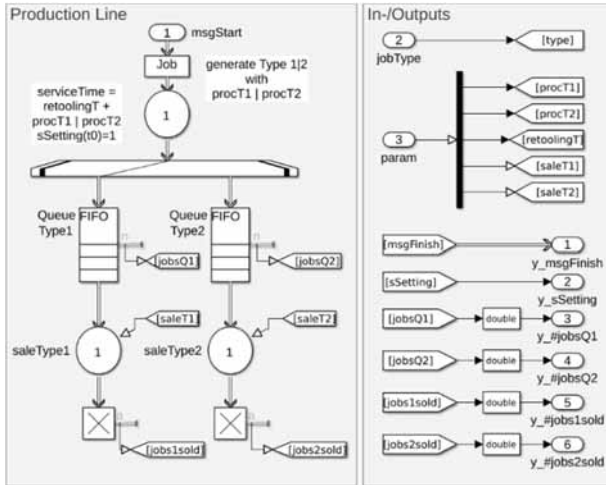


Figure 5. Structure of the MUS in SimEvents.

The calculation of the type of entity and retooling time dependent processing time is done during simulation runtime using two Simulink functions (not shown in Figure 5). After processing, branching into one of the two FiFo queues of the downstream servers takes place depending on the entity type. The downstream servers have different processing times (*saleT1*, *saleT2*). The definition of the different time values is determined by a value vector *param*=[*procT1*, *procT2*, *retoolingT*, *saleT1*, *saleT2*] at input port3 at the simulation start time τ_0 .

Entities are generated via input events (*msgGenJob*) at input port1. The entity type (*jobType*) to be generated follows on from the value at input port2. After an entity has been processed in the operating unit, the MUS generates an output event (*y_msgFinish*) at output port1. Furthermore, the current tool setting (*sSetting*) of the operating unit, the current queue lengths (*y_#jobsQ1*, *y_#jobsQ2*), and the number of completed entities on the downstream servers (*y_#jobs1sold*, *y_#jobs2sold*) are output as data from port2 to port6. Hence, input set I_{MUS} and output set O_{MUS} are defined by:

- $I_{MUS} = \{msgGenJob(\tau), type(\tau), param(\tau_0)\}$
- $O_{MUS} = \{y_msgFinish(\tau), y_sSetting(\tau), y_#jobsQ1(\tau), y_#jobsQ2(\tau), y_#jobs1sold(\tau), y_#jobs2sold(\tau)\}$

The MUS represents the dynamic system behavior independent of a concrete experiment. The goal of the following experiments is to design a controller with the best possible injection strategy of the two entity types into the MUS so then the queues have the most balanced stock of both types available for the downstream servers.

2.2 Designing a Heuristic Strategy

The top-level structure of the DESM for designing a heuristic control strategy is shown in Figure 6. The MUS named *Prodlime* implements the input and output interface described in Section 2.1 with $I_{MUS}(\tau)$ and $O_{MUS}(\tau)$. The I_{EF} / O_{EF} of the EF are not visible on the top-level structure of the DESM.

This interface is realized via workspace variables. The EF consists of five components, of which *Parameters*, *Controller* and *Encoder* form the generator *Gen* according to Figure 2. With the exception of the *Encoder*, the components of the EF operate purely signal-oriented.

The *Transducer* monitors the signal-oriented outputs of the MUS, maps the variables of interest O'_{MUS} for this experiment in a vector $yDec = [sSetting, \#jobsQ1, \#jobsQ2]$, and provides the vector as output variable. Moreover, the *Transducer* generates the SU mapping, by providing the time trajectories of the O'_{MUS} quantities as EF outputs O_{EF} via the data workspace.

The *Acceptor* controls the termination of the simulation after a specified time interval $[\tau_0, \tau_{final}]$ has elapsed. It evaluates the interest variables *#jobsQ1* and *#jobsQ2* and terminates the simulation run abnormally if the difference between the two quantities exceeds a maximum value.

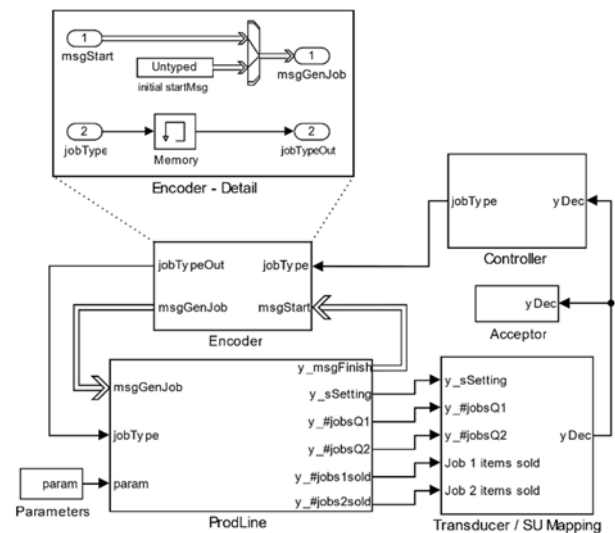


Figure 6. Top-level structure of the DESM with MUS and EF for designing a heuristic control strategy, and substructure of the Encoder block

The *Controller* implements the heuristic strategy. It determines the next entity type to be generated based on the current values of the interest variables received by the *Transducer*.

The goal is to minimize the difference between the two queue contents ($\#jobsQ1$ and $\#jobsQ2$) while respecting the current tool setting ($sSetting$). The result is passed on as a signal ($jobType$) to the *Encoder*.

The *Encoder* works event-driven (see Figure 6). If the operation server of the MUS is free, it sends an event $y_msgFinish$ to the *Encoder*. Thereupon the *Encoder* sends an event $msgGenJob$ to the MUS and forwards the signal $jobType$ that codes the entity type to be generated. At the start of a simulation run, the *Encoder* generates the initial input event $msgGenJob$ and sets the entity type ($jobType=1$) to be generated for the MUS. *Parameters* generates the constant input segments for the MUS vector $param$ for initializing the MUS parameters.

This is a simple SBE. The EC defines the parameters P_{Sim} to be varied and directly calls the SimMeth to execute simulation runs.

2.3 Learning a Strategy Using a Self-Implemented Q-Agent

The top-level structure of the DESM for this SBE is shown in Figure 7. The identical MUS named *Prodline* is integrated into an RL-specific EF according to Figure 4. As in the previous experiment, the EF interface (I_{EF}/O_{EF}) is implemented via Workspace variables. *Parameters*, *Agent* and *Encoder* form the generator *Gen*, and *Decoder*, *Reward* as well as *SU Mapping* form the transducer *Trans* (cf. Figure 2). To learn a strategy, the agent requires unique state-action tuples (s_t, a_t) as well as associated next state s_{t+1} and reward values r_{t+1} . Hence, the two time bases t and τ were introduced in Subsection 1.3 for the EF and the MUS. Accordingly, the components of the EF are implemented event-oriented, with the exception of *Parameters* and *SU Mapping*. The component *Parameters* is identical to the previous experiment.

At simulation start time τ_0 , an episode is started by the *Agent* sending an event $msgGenJob$ and setting an action value $a_t=\{1|2\}$ at the output port $action$. In this case, the outputs of the Agent are compatible with the inputs of the MUS in value and timestamp with respect to the global simulation clock. Hence, the Agent's outputs are only forwarded by the *Encoder* to the MUS *ProdLine* that generates a new entity with $jobType=action$ value.

When an entity has completed on the operation unit, an output event $y_msgFinish(\tau)$ is sent from the MUS to activate the *Decoder* and study-specific $output\ data(\tau)$ is passed signal-oriented to the *SU Mapping* for trajectory recording.

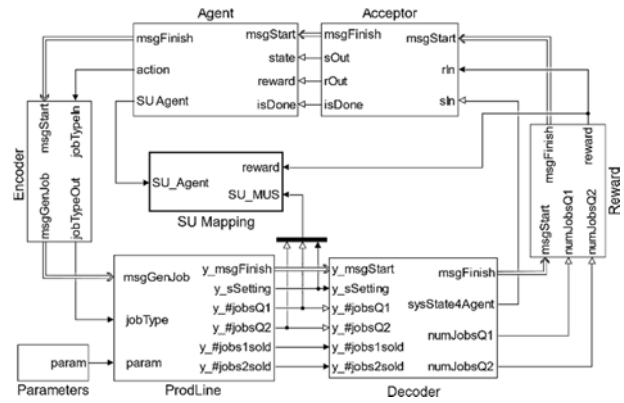


Figure 7. Top-level structure of the DESM with MUS and EF for learning a strategy using a self-implemented agent component.

The *Decoder* selects the information relevant to the RL from the *MUS outputs*(τ) and calculates the new state s_{t+1} of the RL space. To limit the RL space, the Decoder truncates the two queue contents ($\#jobsQ1$ and $\#jobsQ2$) to a maximum length. The new state s_{t+1} is thus calculated from the two limited queue contents and the current tool setting ($sSetting$) of the operating unit, and output at the port $sysState4Agent$.

After decoding, the reward calculation is activated by an event $msgFinish$. Contrary to the general approach, the reward is not calculated using the RL-related state $s \in S$ but on the basis of MUS-related interest variables $O'_{MUS}(t+1)$, in this case $\#jobsQ1$ and $\#jobsQ2$. In terms of content, both approaches are identical but the second one resulted in a much better structured reward computation.

After the reward calculation, the *Acceptor* is activated by an event $msgFinish$. In this experiment, no constraints are defined for s_{t+1} and r_{t+1} , so they are only passed to the *Agent*. Only a control segment is defined for the simulation time interval $[\tau_0, \tau_{final}]$, which specifies the length of an episode. At the termination of an episode, the *Acceptor* schedules an internal event with an infinitesimal time advance. The time delay is necessary for data updates in the *Agent* and *SU Mapping* at the end of an episode. The *Acceptor* activates the *Agent* via an event $msgFinish$ and signals using the boolean variable $isDone$ whether the end of an episode (eof) has been reached or not.

The *Agent* evaluates the boolean $isDone$ value. If $isDone$ is false, it executes its learning rules, calculates a new $action$ value, and generates an output message $msgFinish$ to activate the *Encoder*. In case of $isDone$ is true it performs a data update $SU_Agent(eof)$ and the episode is terminated by the *Acceptor*.

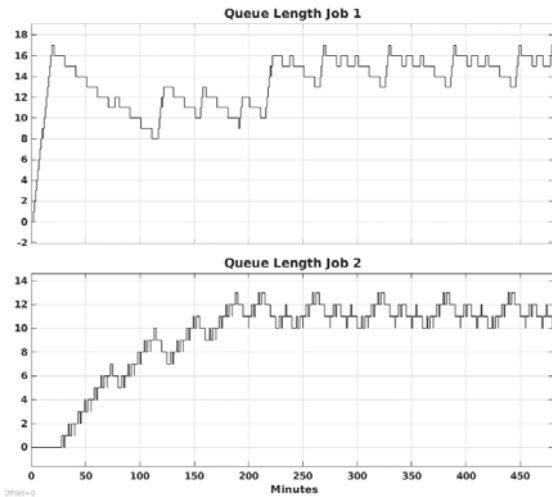


Figure 8. Time trajectories of the queue lengths computed using the learned strategy of a self-implemented Q-agent after 5000 episodes.

This is a *complex SBE*. The EC defines the parameters P_{ExM} and P_{SbM} (cf. Fig. 1), and calls an ExpMeth *training* as described in Subsection 1.3, which calls the SimMeth in a loop to execute the DESM for one episode at a time. Figure 8 shows the simulation results of the MUS using a learned control strategy after 5000 episodes.

The event-oriented implementation of the EF components was done using SimEvents' *Discrete Event Charts*, which call MATLAB functions. This makes the algorithms of the components, such as the learning approach of the agent, easily interchangeable. In Pawletta and Bartelt [6], the algorithms of this experiment are presented in more detail and the full implementation is available on Github (FG CEA [4]).

2.4 Learning a Strategy Using a Dedicated RL Toolbox

The MathWorks offers a dedicated RL toolbox for MATLAB/Simulink (MathWorks [10]). This provides an agent block for Simulink, which is configured from MATLAB. Different learning approaches can be configured in the form of agent types as well as hyperparameter settings. Furthermore, the toolbox provides different methods, such as a training method called *train*. In this experiment we use the Q-learning agent and the *train* method of the RL toolbox. It must be noted that the documentation of the toolbox does not contain any hints or examples for the use with event-oriented MUS implemented with the SimEvents blockset. According to the documentation, the agent block of the RL toolbox works signal-oriented.

A signal is in Simulink a time-varying quantity that has values at all points in time. Accordingly, the agent block is designed for continuous or discrete-time models with equidistant sampling.

The top-level structure of the DESM for this SBE is shown in Figure 9. With the exception of the *Triggered Agent* block, the DESM corresponds completely to the DESM in Figure 7, i.e., all other components of the EF as well as the MUS were adopted unchanged. Hence, only the *Triggered Agent* is discussed below.

The *Triggered Agent*, implements a so-called triggered subsystem and encapsulates the RL agent block of the toolbox. The inner structure of the *Triggered Agent* and the input/output interface of the encapsulated RL agent are also shown in Figure 9. Analogous to the *Agent* in the previous model (cf. Figure 7), the *Triggered Agent* is activated by the *Acceptor* per event (*msgStart*) when a new state s_{t+1} of the RL space (input port *sIn*) and a new reward value (input port *rIn*) as well as the boolean *isDone* value have been calculated. If *isDone* is false, the encapsulated RL agent calculates the next *action value a*, as well as updates the *cumulative reward value*, and then the *Triggered Agent* activates the *Encoder* by event (*msgFinish*).

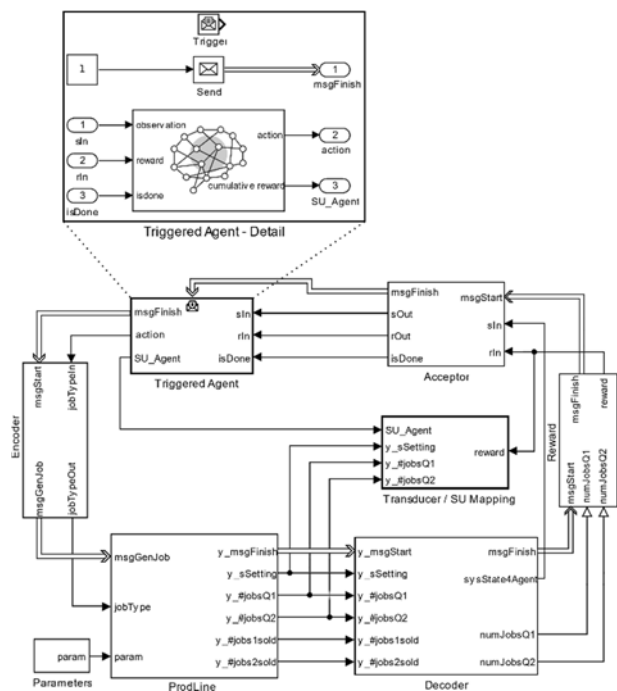


Figure 9. Top-level structure of the DESM with MUS and EF for learning a strategy using the RL Toolbox of MATLAB/Simulink, and substructure of the *Triggered Agent*.

In the other case, if *isDone* is true, the encapsulated RL agent computes the complete data update of the episode $SU_Agent(eoe)$ and, in contrast to the previous model, immediately terminates the episode. I.e., the termination of the episode by the acceptor according to the previous example is skipped.

The EC defines the parameters P_{ExpM} and P_{SimM} , and calls the RL toolbox specific ExpMeth *train*, which uses an RL toolbox specific SimMeth.

3 Conclusions

The integration of discrete-event simulation and RL methods has a high application potential for both M&S and AI applications. On the basis of the concept of EF and the general structure of complex SBE, it has been shown how a clear methodological separation can be made so that the MUS, EF, SimMeth, simulator and AI methods – as ExpMeth – can be developed independently and reused in different contexts. The methodological considerations have been practically underpinned by a case study implemented with MATLAB/Simulink and the SimEvents blockset.

In particular, the three experiments of the case study demonstrate that MUS can be developed independently of their experimental context. As shown, this is also true when integrating with the RL method. The adaptation to a concrete experiment can be done by a specific EF, higher-level ExpMeths and a supervisory EC. The basic structure of an EF and the communication relationships in SBE using the RL method were presented.

SBEs in combination with the RL method are characterized by a large number of methodological parameters and variants of agents. Accordingly, the specification of such experiment variants and their automated execution based on the System Entity Structure and Model Base (SES/MB) approach will be investigated in a next step.

References

- [1] Breitenacker F. Models, methods and experiments - a new structure for simulation systems. *Mathematics and Computers in Simulation*, 1992, 34(3):231–260.
- [2] Carson Y, Maria A. Simulation optimization: methods and applications. In *Proceedings of the 1997 Winter Simulation Conference*, 1997, 118-126.
- [3] Choo B, Graham C, Stephen A, Dadgostari F, Beling PA. Reinforcement learning from simulated environments: an encoder decoder framework. In *Proceedings of the SCS SpringSim'20 (Virtual) Conference*, 2020. 12 pages.
- [4] FG CEA. Integration of RL and Discrete Event Simulation: A Case study using MATLAB/ Simulink/ SimEvents, Wismar Univ. of Applied Sciences Wismar, 2022, <https://github.com/cea-wismar>.
- [5] Pawletta T. Specification and execution of simulation models and experiments. *Discussion talk at MS Workshop 'One simulation model is not enough'*, Univ. of Rostock, Dep. of Computer Science, (2019) https://www.cea-wismar.de/pawel/Forschung/Poster_Slides/2019-04-23-Presi_FG-CEA_UnivRo-WS_reducedSize.pdf.
- [6] Pawletta T, Bartelt J. Integration of Reinforcement Learning and Discrete Event Simulation Using the Concept of Experimental Frame. In Mota M.M., editor. *Eurosim Congress 2023*; 2023 Jul; Amsterdam, Netherlands. 8 pages (submitted 2022-Dec-30).
- [7] Schmidt A. Variantenmanagement in der Modellbildung und Simulation unter Verwendung des SES/MB Frameworks (Variant Management in Modeling and Simulation using the SES/MB Framework). *Advances in Simulation – Fortschrittsberichte Simulation*, Bd. 30, ARGESIM Publisher, Vienna. ISBN ebook 978-3-903347-30-4. DOI 10.11128/fbs.30
- [8] Sutton RS, Barto, AG. *Reinforcement Learning: an Introduction – 2nd edition*. 2018, MIT Press.
- [9] The MathWorks (2022-1). SimEvents. <https://mathworks.com/simevents/reinforcement-learning.html>, ©1994-2022 The MathWorks, Inc.
- [10] The MathWorks. Reinforcement Learning Toolbox. (2022-2); <https://mathworks.com/products/reinforcement-learning.html>, ©1994-2022 The MathWorks, Inc.
- [11] Traore MK, Muzy A. Capturing the dual relationship between simulation models and their context. *Simulation Modeling Practice and Theory*, Elsevier, 14(2006), 126-142.
- [12] Zeigler BP. *Theory of Modeling and Simulation – 1st edition*. 1976, John Wiley & Sons.
- [13] Zeigler BP. *Multifaceted Modelling and Discrete Event Simulation*. 1984, Academic Press.
- [14] Zeigler BP, Muzy A, Kofman E. *Theory of Modeling and Simulation – 3rd edition*. 2018, Elsevier, Academic Press.

Implementing Thermodynamic Cyclic Processes Using the DLR ThermoFluid Stream Library

Peter Junglas*

PHWT-Institut, PHWT Vechta/Diepholz, Am Campus 2, 49356 Diepholz, Germany;

*peter@peter-junglas.de

SNE 33(4), 2023, 175-182, DOI: 10.11128/sne.33.sw.10665
Selected ASIM SST 2022 Postconf. Publication: 2023-08-15;
Rec. Revised Improved: 2023-11-23; Accepted: 2023-11-24
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. Simulation programs modeling cyclic processes can be used in thermodynamics lectures to promote understanding. Modelica-based simulation environments are a good starting point for the development of such programs, but the handling of the corresponding thermo-fluid standard library is very difficult for non-experts. The recently presented DLR ThermoFluidStream Library is a good alternative, that is easier to use. It provides most components that are needed in typical cases and includes full access to the Modelica media library. It will be shown, how to use the ThermoFluidStream Library to create examples ranging from the simple Otto and Diesel cycles over the basic Joule-Brayton and Ericsson processes to the water/steam based Clausius-Rankine cycle. Though the construction of concrete processes with given thermodynamic state values and mass flow still requires some effort, one can apply a systematic approach for working models for teaching purposes.

1 Introduction

Thermodynamics is a difficult subject for many engineering students, mainly because of its partly unintuitive nature using abstract notions like enthalpy and entropy. To promote understanding simulation programs can be used, which allow to “experiment” with state changes or complete cyclic processes, such as the collection of Java applets described in [1].

But the construction of such programs is a tedious and time-consuming task, especially if one wants to include examples that use more complex media than the simple ideal gas with constant specific heat capacity.

Instead of writing such programs from scratch, one could use a simulation environment to describe the example models, and leave the actual computation to its solver. Modelica [2] with its physical modeling approach seems to be a good starting point, especially since a comprehensive free model library is available that describes the thermodynamic behaviour of many useful media [3]. Therefore it will be used in the following to build models of the standard processes that are examined in most introductory thermodynamics courses: the Otto and Diesel processes for closed systems and the Joule-Brayton and Ericsson processes for open systems [4, 5]. These models should run on any Modelica platform, especially on the freely available OpenModelica environment [6], and can be employed directly in a thermodynamics course.

The Modelica Standard Library (MSL) already contains an elaborate thermo-fluid library that provides basic components for one-dimensional thermo-fluid flow in pipes, vessels or machines [7]. But due to its very general approach it is much too complicated for the simple didactical applications addressed here. Additionally, corresponding models recurrently fail to run for reasons that are hard to find for non-specialists [8].

The recently presented DLR ThermoFluidStream Library [9] (“ThermoDLR”) seems to provide just the level of detail that is needed here: On the one hand it uses the full Modelica media library, on the other hand it offers components for vessels and machines that are much easier to handle than their MSL counterparts. And, most importantly, it uses a very clever, physically motivated scheme to achieve a high robustness [10] that should lead to models that generally run without delicate fine-tuning. This makes it a promising foundation for the construction of didactical examples.

A similar, but simpler and limited approach has been presented in [11], which also describes a Modelica library for thermodynamical examples (“ThermoSimT”).

Since its focus is on teaching Modelica, it does not use the complex standard Media library, but a greatly simplified version. Especially its steam/water model only shows basic modeling principles, but is of no practical use. Nevertheless it allows to easily build models of all standard processes and will be used as a benchmark to assess the ease of use and versatility of the new ThermoDLR version.

In the following we will briefly describe the basic ideas of the ThermoSimT library, which provides models of cyclic processes for ideal gas with constant or temperature-dependent specific heat capacity and of the steam/water based Clausius-Rankine cycle. Then we will use the ThermoDLR library to implement similar models, utilizing the Media library to get valid practical results. Due to the didactical purpose and since we are only interested in equilibrium behaviour, the models have some unusual features: Heat transfer is done very fast, the characteristics of compressors or turbines don't really matter, and the values of the mass flow and several state variables are given in advance.

We will show, which problems appeared during the implementation, and present ways how to deal with them. This will help to produce similar models for own teaching purposes. As a starting point, all models described here can be downloaded freely from [12].

2 Cyclic Processes in ThermoSimT

Since [11] is a textbook on modeling and simulation, the main purpose of the ThermoSimT library is to teach the design and construction of a Modelica library. But thermo-fluid modeling is a very difficult task, therefore a lot of simplifying assumptions had to be made: The mass flow is constant and the flow has always the same direction, i. e. all connections are uniquely defined as input or output ports. The components have no states describing an internal change, but the thermodynamic variables just jump from the input to the output state. As a consequence, the described models are static, time changes can only be implemented by changing work or heat flows.

Since stream connectors [13] are much too advanced for an introductory textbook, the connector is based on the preliminary version of the thermo-fluid library described in [3]. ThermoSimT contains components for simple devices such as a cylinder, a heater, a pump and a turbine, together with source and sink components

and a state measurement device that outputs all relevant thermodynamical variables.

Pump and turbine are identically modeled as simple turbo machines based on an isentropic state change with a simple linear characteristic

$$\dot{m} = K \omega.$$

The simple Media library covers the ideal gas with constant heat capacity ("simple air"), the NASA dry air model [14] and a simple model for steam and water, using ideal gas and ideal fluid equations together with a Clausius-Clapeyron based vapor pressure curve.

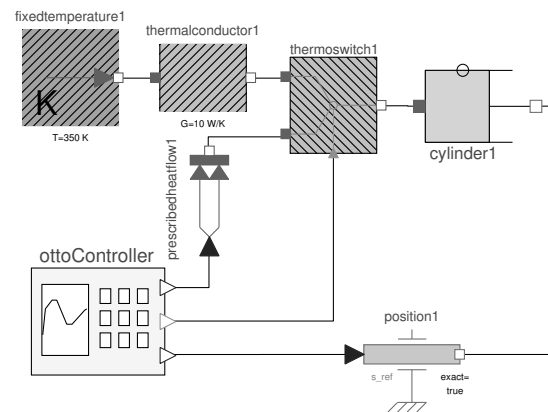


Figure 1: Otto cycle using ThermoSimT.

With these components a model of an Otto cycle can be built easily (cf. Figure 1). The thermodynamic computations are done in the cylinder component, while additional blocks provide a test stand defining the position of the piston and the amount of external heat. Models using simple air and dry air are provided, as well as a similar example for the Diesel process.

Much more interesting from a modeling perspective is the model of the Joule-Brayton cycle (cf. Figure 2), which describes the actual flow of the medium between components. The mechanical work for the compressor is provided by a constant torque block, while the external load at the turbine is modeled using a simple generator and a resistor. The ThermoSimT library does not work with a cyclic topology, but an additional cooler at least brings the state of the medium back to its initial state. Versions for simple air and dry air are provided, as well as an identically looking model using the simple steam/water medium, which actually makes it a Clausius-Rankine cycle.

The final example is the Ericsson cycle, which contains several compressors and coolers as well as turbines and heaters, to approximate an isothermal behaviour in the turbo machines. It can be modelled easily with ThermoSimT (cf. Figure 3).

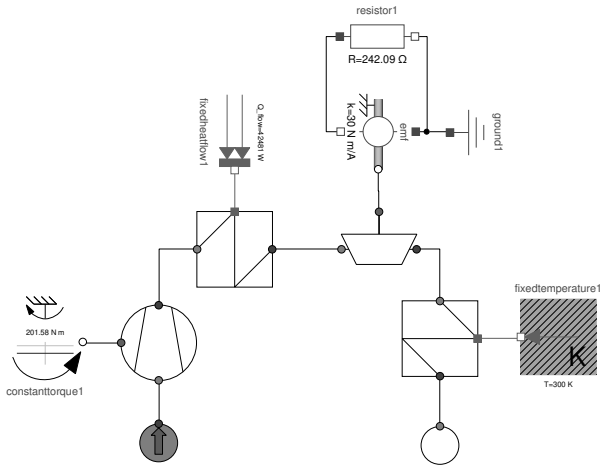


Figure 2: Joule-Brayton cycle using ThermoSimT.

3 Otto and Diesel Processes

Since a cylinder model, which is at the heart of the Otto cycle model, is not included yet in the ThermoDLR library, one has to build it oneself. Fortunately, a complete volume model already exists, together with several variants. They all inherit from the parent class `PartialVolume`, which provides most of the variables and equations needed, together with optional `HeatPort`, `Inlet` and `Outlet`. To construct a `CylinderVolume`, one simply extends `PartialVolume`, adds a mechanical `Flange` and provides simple equations for the definition of the volume, the force and the work at the flange. To simplify the drawing of a T-s diagram, an explicit variable for the entropy is added.

Exchanging the cylinder model in the ThermoSimT Otto cycle is all that remains to do. The new model works immediately with `DryAir`, for `SimpleAir` one has to extend its range of validity by defining

```
SimpleAir(T_min=200, T_max=2000)
```

The Diesel controller component needs the internal pressure to create an isobaric process, which can be supplied by a sensor at the optional `Outlet` of the `CylinderVolume`.

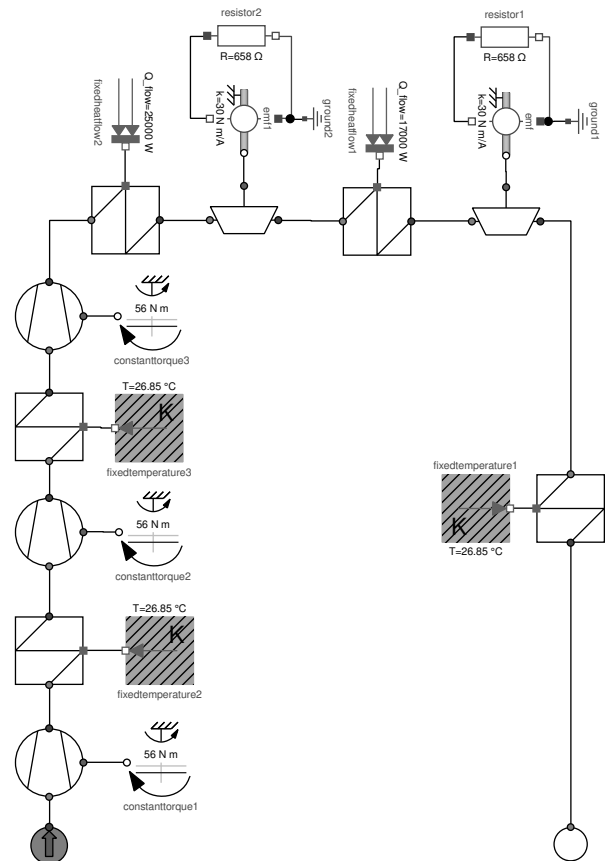


Figure 3: Ericsson cycle using ThermoSimT.

4 Joule-Brayton Process with Ideal Gas

The construction of a model for the Joule-Brayton cycle seems to be almost trivial. One starts by replacing the thermodynamic components from the ThermoSimT example with their counterparts from ThermoDLR: `Source`, `Sink`, `Compressor`, `Turbine`, and `ConductionElement` as replacement of the `Heater`. All parameters have their default values except for the initial pressure and temperature at the `Source`. That the mass flow can not be defined anywhere, is due to the different concept of ThermoDLR: \dot{m} is a dynamic variable and is computed in the context of the whole model – here probably mainly depending on the compressor parameters. With the ideas from ThermoSimT in mind, this seems to be strange, and indeed: Starting the simulation results in the infamous error “Failed to solve nonlinear system using Newton solver during initialization.”

To find the reason of this problem, one examines a simple test model consisting of a `Source`, a `Compressor` and a `Sink`. The `Source` defines starting pressure $p_1 = 1$ bar and temperature $T_1 = 300$ K, the `Sink` the final pressure $p_2 = 6$ bar. Using the same torque as in the `ThermoSimT` model, the simulation doesn't run, which is not surprising, since the compressor from `ThermoDLR` uses a completely different characteristic curve and default operating point.

To find an adequate torque value, one has to analyze the equations used in the `Compressor` component. Using default parameters and neglecting regularisations for small variable values, they are:

$$\frac{p_2}{p_1} = \frac{\omega^2}{\omega_{ref}^2} - \frac{\dot{m}^2}{\dot{m}_{ref}^2} + 1$$

$$w_t = \frac{\kappa}{\kappa - 1} RT_1 \left(\left(\frac{p_2}{p_1} \right)^{\frac{\kappa - 1}{\kappa}} - 1 \right)$$

$$= \frac{\omega \tau}{\dot{m}}$$

They define a quadratic pressure characteristic and compute the work using the explicit formula of an isentropic process for an ideal gas with constant heat capacity. Inserting the given state variables, the desired mass flow $\dot{m} = 0.1$ kg/s and the default parameter values, one easily computes values for ω and τ .

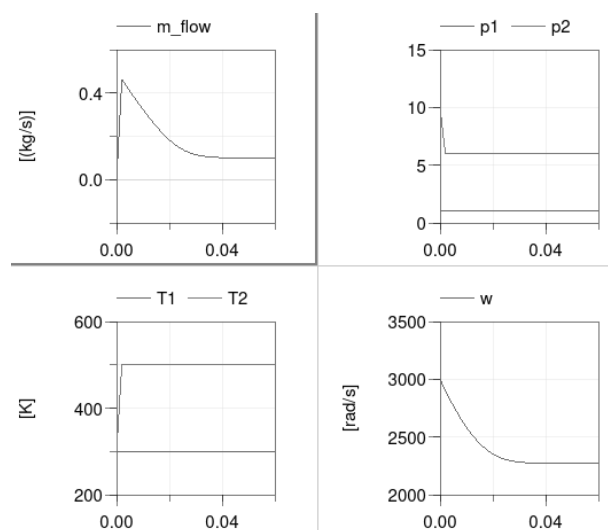


Figure 4: Results of the compressor test model.

Using this torque value for the constant torque and setting the initial value of the rotational velocity to ω (at least approximately), the test model runs and has

the correct results (cf. Figure 4). In accordance to the philosophy of the `ThermoDLR` library, the initial value of \dot{m} is chosen to be 0, so that the equilibrium is reached by simulating the powering up of the system.

A similar computation can be done for the turbine. Unfortunately, the identical characteristic curve of compressor and turbine contains a very strict regularisation, whenever the pressure drops (as in the turbine). But of course it's simple to create a copy of the curve model and use the quadratic characteristic for the turbine as well. Trying to compute the torque for the given state variable values, one gets a negative value under a square root. This problem can easily be fixed by changing the operation point, setting $\dot{m}_{ref} = 0.05$ kg/s. Finally one utilizes the linear characteristic of the simple generator to compute the resistance R , and gets the requested results.

For the heater, no new computations are necessary, since the needed heat is fixed by the thermodynamics. Combining the components (for a start without the final cooler) using the new parameter values, one gets a working Joule-Brayton process, which almost reproduces the required values. Only the pressure p_2 after the compressor is slightly higher in the combined model, maybe due to the effect of several regularisations.

Instead of going through the complete equations, the fine-tuning can easily be done with a few manual parameter changes: First one lowers the torque at the compressor, until p_2 reaches the desired value. Here, the `MultiSensor` components are very convenient, which display state variables directly in the graphical model. Now all pressure and temperature values are correct, only the mass flow is a bit too small. Better than fiddling with several parameters at once, one can use a simple scaling procedure: Increasing all works and heats by a constant factor q one can change only the mass flow, not the thermodynamical state. Setting $q = \dot{m}_{desired} / \dot{m}_{actual}$, increasing \dot{m}_{ref} , τ and \dot{Q} by a factor q and dividing R by q , one eventually reproduces all `ThermoSimT` values.

Finally one adds another `ConductionElement` with a large heat transfer coefficient to bring the temperature down to the start value. With the `ThermoDLR` library, one can now make it a real cycle by deleting the `Sink` and `Source` components and closing the circle with an intermediary `Volume` element. The initial state, which had been defined by the `Source`, is now given as initial value of the `Volume`.

5 Ericsson Process with Ideal Gas

To simplify the construction of an Ericsson model, a subsystem is built for a line of three compressors and intermediate coolers (cf. Figure 5), as well as a similar model of two turbines with intermediate heater. Both components contain sensors to output the complete work and heat supplied to the line. This will be useful to compute the overall efficiency in the complete model.

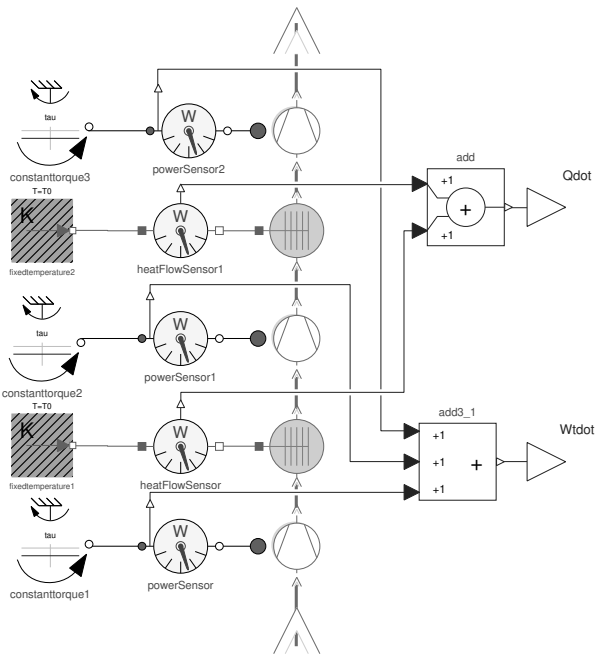


Figure 5: Line of compressors for the Ericsson cycle.

Providing correct parameter values, these compressor and turbine components show approximately isothermal behaviour. Using them in the Joule-Brayton model and guessing reasonable values for the work-related parameters τ and R , the Ericsson model runs – at least, if one lowers the intermediate heating temperature a bit. As before, one can use fine-tuning to get the given pressure values and scaling to reach the correct mass flow. The resulting model is more stable, so that one can raise the intermediate heating temperature to the incoming value to better approximate an isothermal process.

Now we can make good use of the additional possibilities that are supplied by ThermoDLR, and include a heat exchanger that utilizes excess heat after the tur-

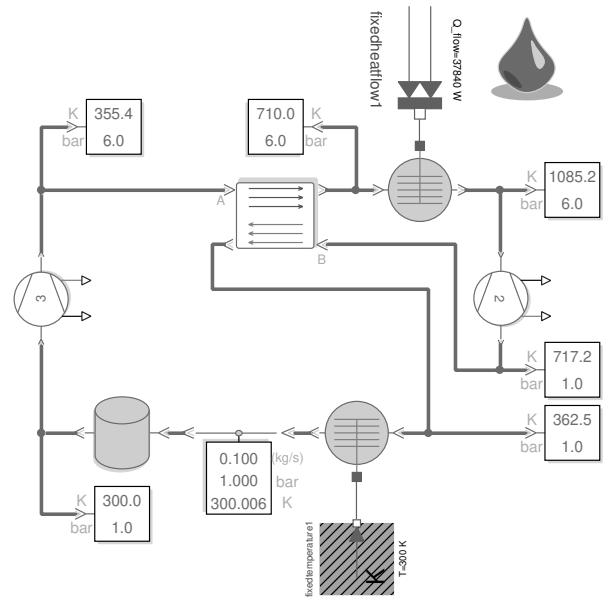


Figure 6: Ericsson cycle with heat exchanger using ThermoDLR.

bine for preheating the gas before entering the heater (cf. Figure 6). To reach the same maximal temperature as before, one starts with a low heat transfer coefficient k_{NTU} , then gradually lowers the supplied heat, while rising k_{NTU} . The final Ericsson model is much more realistic than the previous version based on ThermoSimT.

6 Joule-Brayton Process with Dry Air

Building a Joule-Brayton example based on dry air as a medium should be done easily by simply changing SimpleAir to DryAirNasa, but unfortunately, the new model doesn't run: To compute the temperature from the given enthalpy, the function $h(T)$ has to be inverted – but the Dekker-Brent based solver is supplied with a start interval without a zero.

Looking closer at the DryAirNasa implementation, one finds another problem: The computation of the isentropic enthalpy is based on the approximation of constant heat capacity. This may be good enough for many practical applications, but in a teaching context one should present a correct solution. Since the utilised function `isentropicEnthalpy` is not declared as `replaceable`, one has to create a copy of the `DryAirNasa` class and its base class `SingleGasNasa` and supply a computation that

is based on the constant entropy. Additionally, the function `dp_tau_const_isentrop` that is used in ThermoDLR to define the compressor characteristics, is based on a constant heat capacity formula as well and has to be replaced by an exact version `dp_tau_const_isentrop_S`.

Using these more accurate methods, one is still faced with the original problem: The inversion gets a wrong start interval. This is due to the initial value $\dot{m}(0) = 0$: The direction of the flow is undefined initially and the algorithm tries values that are going into the wrong direction. Simply setting $\dot{m}(0) = 0.1$ kg/s, which is given anyhow, succeeds and the model runs – but the values of pressure and mass flow are not the required ones. Obviously, the seemingly small difference between `SimpleAir` and `DryAir` leads to larger deviations than expected.

To find correct parameter values, one again starts with a simple test model for the compressor. Either by a direct computation – which can be easily done for dry air with a small Matlab script – or by a few trial and error steps, one finds values that lead to the required pressure and mass flow. Due to the internal inversions, the model is less stable than its `SimpleAir` counterpart: Even if one sets the correct torque value, one still has to supply an initial value for ω that is large enough. For simplicity one can instead directly define ω at the flange. The corresponding computation for the turbine again shows that \dot{m}_{ref} has to be lowered. After that, the computation leads to the correct value of the resistance R . Closing the cycle with a cooler and a volume element, one arrives at a Joule-Brayton model with dry air that – after a bit of the usual fine-tuning – reproduces the given state and mass flow values.

7 Clausius-Rankine Process with Standard Water

Basically, the Clausius-Rankine process is a Joule-Brayton process with water and steam as a medium: Though its technical realisation is much more complicated due to the phase transitions from water to steam and back, conceptually it consists of a pump, a heater, a turbine and a cooler. The media library includes `StandardWater`, a precise description of water and steam based on the IAPWS-IF97 formulation [15]. Yet, one cannot just use one of the Joule-Brayton models and change the medium, since the compressor and turbine components of ThermoDLR use explicit ideal gas

relations. For incompressible fluids the `Pump` component can be used, a suitable turbine component is not provided in the library. Also, completely different values of the state variables than before will be employed: The pressure ranges from 0.1 bar to 60 bar, the highest temperature should be 500 °C and the mass flow 10 kg/s.

To find working parameters, one can start with a simple test model for the pump. The ThermoDLR `Pump` component provides two different characteristics: a centrifugal pump and a simpler nominal pump, which is used in the following. Setting reasonable nominal values and basic parameters, one can once again use the model equations and the given state and mass flow to compute a correct value of the rotational speed ω . Adding a standard heater (i. e. a `conductionElement`), one can reach the given temperature by first estimating, then fine-tuning the needed heat flow. The heating process includes the complete vaporisation of the water, succeeded by an overheating of the steam, but all this is automatically taken care of by the state equations used in IF97. Adding corresponding sensors, one can easily monitor the dryness fraction x everywhere in the cycle model.

A generic turbine model `TurbineG` that works for `StandardWater` (and any any other fluid medium) can be easily constructed: It inherits from the provided `PartialTurboComponent` and uses the explicit characteristic function `dp_tau_const_isentrop_S` that has already been defined for the `DryAir` example. Since the medium changes its phase from hot steam to wet steam inside the turbine, one can expect numerical difficulties, and in fact: Using default parameter values, the usual test model doesn't run, because the pressure reaches values below the triple point of water. To find working values, one initially starts with a simpler process, going from 60 bar to 30 bar instead of trying to reach 0.1 bar immediately. Changing ω_{ref} und \dot{m}_{ref} , one soon gets a model that at least runs for a very short time. This makes it possible to study its behaviour and find the reason of the problems. Adapting ω , the model finally runs to the end. Now one can use the usual fine-tuning to gradually lower the pressure to the requested value and to switch from the constant ω input to the simple consumer model. The needed resistance value is completely unrealistic, but can finally be scaled to a meaningful value by changing the transformation coefficient of the simple generator model.

Now a first version of a Clausius-Rankine cycle can be built by combining the pump-heater and turbine models. The model runs only for a short time, before the final pressure again drops below the triple point. Enlarging the value of R by a factor of 100, the model runs to the end, though the state values are far from the requested ones. Once again one goes through the fine-tuning process, changing ω , \dot{Q} and R , to reach the requested state values. This is much more tedious than in the previous examples, because the very different properties of water and steam lead to a small range of working parameters. Additionally, their strong coupling sometimes results in a counterintuitive behaviour, as in the following scenario: One finds a too low temperature behind the heater and increases the heat flow, which leads to an increase of the mass flow and a falling temperature. Another problem is the instability of the standard DASSL solver that is used in the simulation program Dymola: Occasionally, the solver hangs for certain parameter values and has to be stopped. Changing to the ESDIRK23a solver drastically reduced the frequency of such events. With a certain amount of perseverance, results within 1 % of the requested values could be reached.

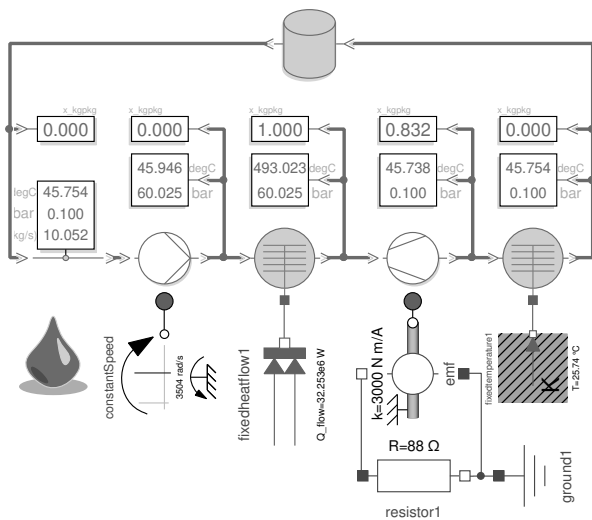


Figure 7: Clausius-Rankine cycle using ThermoDLR.

The next step is the addition of the cooler, which is supplied with a fixed temperature. Choosing the obvious value of 45.80°C – the condensation temperature at 0.1 bar –, one doesn't reach $x = 0$, therefore one has to use a smaller value and to increase the heat transfer coefficient U considerably. Finally adding the volume

element to close the circle, one again gets very different results and has to tune the cooler parameters once more, as well as the initial temperature of the volume. But in the end one reaches a complete Clausius-Rankine cycle with the requested state values (cf. Figure 7).

8 Conclusions

All examples of the simple ThermoSimT library could be implemented in ThermoDLR, as well as more complex processes using a heat exchanger or an accurate water model. ThermoDLR is designed to create models that run immediately, nevertheless it was nontrivial to build models that reproduce the given state values and mass flows. A systematic approach often worked that was based on the following steps:

- Start with simple models, where initial and final states are defined by `Source` and `Sink` components.
- Change the operation points of turbo machines to values near the given state values.
- Use (simplified) versions of the component equations to get good starting points for external parameters.
- Gradually fine-tune parameters to approximate the given thermodynamic states. If necessary, change the mass flow by a scaling procedure.
- Combine partial models and reiterate the fine-tuning process.
- Finally close the cycle with a `Volume` component and properly define its initial values.

Sometimes it was difficult to find proper parameters to make the model run at all or to change the state into the required direction – especially for a complex medium such as water and steam. In such cases, it proved useful to set the initial value of the mass flow directly to the required value or to define operation points very close to the target values. Generally, such a model runs at least for a very short time. This is helpful, because seeing the results of small parameter changes directly can provide clues on what to do to finally reach a running model. Other helpful measures were the introduction of control valves to decouple parts and define intermediate states, or even to change the solver. When the model finally works as required, it is generally more

stable than the previous versions and allows for easy parameter changes.

The design of the ThermoDLR library, and especially its ability to start with zero mass flow and model a power-up situation, makes the modeling of thermo-fluid systems much simpler for non-specialists than the complex Modelica Thermo-Fluid library. On the other hand, the consistent use of the mass flow as a state variable sometimes leads to unexpected behaviour, when parameters are changed or submodels combined. ThermoDLR provides useful basic components that are easy to understand down to the equation level and are easy to extend due to a simple, but convenient inheritance hierarchy. The supplied `MultiSensor` components are very helpful during the fine-tuning phase. Some important elements are missing yet, such as a generic turbine with a corresponding characteristic function or a cylinder volume, and had to be added here.

On the whole, ThermoDLR proved to be a very useful tool for the construction of cyclic process models that can be used for demonstration purposes in thermodynamics lectures. Of course, it requires some effort to familiarise oneself with the library, but going in simple steps, as has been shown before, the learning curve is not very steep and can be mastered by lecturers, who are not experts in thermo-fluid modeling.






Acknowledgement

The author is grateful to Dirk Zimmer, Niels Weber and Michael Meißner for introducing him to the DLR ThermoFluid Stream Library and for providing helpful hints during the construction of the described models.

References

- [1] Junglas P. Simulation Programs for Teaching Thermodynamics. *Global J of Engng Educ.* 2006; 10(2):175–180.
- [2] Modelica Association. *Modelica® – A Unified Object-Oriented Language for Systems Modeling, Language Specification Version 3.5.*
URL <https://modelica.org/documents/MLS.pdf>
- [3] Casella F, Otter M, Proelss K, Richter C, Tummescheit H. The Modelica Fluid and Media library for modeling of incompressible and compressible thermo-fluid pipe networks. In: *Proc. 5th Int. Modelica Conference.* 2006; pp. 631–640.
- [4] Moran MJ, Shapiro HN, Boettner DD, Bailey MB. *Fundamentals of Engineering Thermodynamics.* Hoboken, NJ: John Wiley & Sons, 9th ed. 2020.
- [5] Cerbe G, Wilhelms G. *Technische Thermodynamik.* München: Carl Hanser, 19th ed. 2021.
- [6] Fritzson P, Pop A, et al. The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development. *Modeling, Identification and Control.* 2020;41(4):241–295.
- [7] Franke R, Casella F, Sielemann M, Proelss K, Otter M. Standardization of thermo-fluid modeling in Modelica.Fluid. In: *Proc. 7th Int. Modelica Conference.* Como, Italy. 2009; pp. 122–131.
- [8] Drente P, Junglas P. Simulating a simple pneumatics network using the Modelica Fluid library. *SNE Simulation Notes Europe.* 2015;25(2):85–92.
- [9] Zimmer D, Weber N, Meißner M. The DLR ThermoFluidStream Library. In: *Proc. 14th Int. Modelica Conference.* Linköping, Sweden. 2021; pp. 225–234.
- [10] Zimmer D. Robust object-oriented formulation of directed thermo-fluid stream networks. *Mathematical and Computer Modelling of Dynamical Systems.* 2020; 26(3):204–233.
- [11] Junglas P. *Praxis der Simulationstechnik.* Haan-Gruiten: Verlag Europa-Lehrmittel. 2014.
- [12] Junglas P. *Thermodynamic Cyclic Processes library in Modelica.*
URL <http://www.peter-junglas.de/fh/simulation/thermocycle.html>
- [13] Franke R, Casella F, Otter M, Sielemann M, Elmqvist H, Mattson SE, Olsson H. Stream connectors – an extension of Modelica for device-oriented modeling of convective transport phenomena. In: *Proc. 7th Int. Modelica Conference.* 2009; pp. 108–121.
- [14] McBride BJ, Zehe MJ, Gordon S. NASA Glenn Coefficients for Calculating Thermodynamic Properties of Individual Species. *Nasa report tp-2002-211556,* NASA. 2002.
- [15] Wagner W, Cooper JR, et al. The IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam. *J Eng Gas Turbines and Power.* 2000;122(1):150–182.

Iterative Scenario-Based Testing in an Operational Design Domain for Artificial Intelligence Based Systems in Aviation

Bojan Lukić^{1*} , Jasper Sprockhoff¹ , Alexander Ahlbrecht¹ , Siddhartha Gupta¹ ,
Umut Durak¹ 

¹German Aerospace Center (DLR), Institute of Flight Systems, Lilienthalplatz 7, 38108 Braunschweig, Germany

*bojan.lukic@dlr.de

SNE 33(4), 2023, 183-190, DOI: 10.11128/sne.33.tn.10666
Selected ASIM WS 2023 Postconf. Publication: 2023-10-15
Rec. Revised Improved: 2023-11-30; Accepted: 2023-12-06
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. The development of Artificial Intelligence (AI) based systems is becoming increasingly prominent in various industries. The aviation industry is also gradually adopting AI-based systems. An example could be using Machine Learning algorithms for flight assistance. There are several reasons why adopting these technologies poses additional obstacles in aviation compared to other industries. One reason is strong safety requirements, which lead to obligatory assurance activities such as thorough testing to obtain certification. Amongst many other technical challenges, a systematic approach is needed for developing, deploying, and assessing test cases for AI-based systems in aviation.

This paper proposes a method for iterative scenario-based testing for AI-based systems. The method contains three major parts: First, a high-level description of test scenarios; second, the generation and execution of these scenarios; and last, monitoring of scenario parameters during scenario execution. The scenario parameters, which can be for instance environmental or system parameters, are refined and the test steps are executed iteratively. The method forms a basis for developing iterative scenario-based testing solutions.

As a domain-specific example, a practical implementation of this method is illustrated. For an object detection application used on an airplane, flight scenarios, including multiple airplanes, are generated from a descriptive scenario model and executed in a simulation environment. The parameters are monitored using a custom Operational Design Domain monitoring tool and refined in the process of iterative scenario generation and execution. The proposed iterative scenario-based testing method helps in generating precise test cases for AI-based systems while having a high potential for automation.

Introduction

The practical use of Machine Learning (ML) applications for Artificial Intelligence (AI) based systems in aviation is still in an early stage. One reason is the premature nature of guidelines illustrating the proper implementation of those applications. Specifically, the additional and strict requirements and constraints for introducing new systems in the aviation industry pose an obstacle. This makes the implementation and certification of ML algorithms for autonomy challenging. Recently, the European Union Aviation Safety Agency (EASA) [1] and Society of Automotive Engineers (SAE) [2] each published early versions of fundamental guidelines, discussing the implementation of ML applications in aeronautical systems. These guidelines provide requirements to support the integration of ML-enabled sub-systems and guidance for implementing ML applications. Due to the premature nature of these guidelines, the certifiability of ML applications in aviation, especially for fully AI-based systems, is not yet given. Yet, similar to traditional software, it is certain that specific verification artifacts need to be provided to increase trust. Typical artifacts include the results of conducted tests.

As defined in the EASA guidance, implementing AI-based systems requires the exact definition of their Operational Design Domain (ODD). The ODD defines the conditions under which a system operates correctly. Specifically, the ODD outlines the operating parameters, encompassing the range and distribution in which the AI/ML component is intended to function. It will function as intended only when the specified parameters within the ODD are met.

Moreover, the ODD takes into account interdependencies among operating parameters to adjust the ranges as necessary. This means that the ranges for one or more operating parameters may be contingent upon the value or range of another parameter [1, 3]. The definition of parameter boundaries for the correct behavior of an AI-based system becomes especially important when working in safety-critical domains such as aviation. For instance, the ODD of an aviation system can help with the definition of design assurance levels [4]. In this context, the definition of the system's ODD supports the generation of precise test cases for high test coverage.

One systematic approach for developing test cases for AI-based systems in their operational domain is model-based testing using the Model-Based Systems Engineering (MBSE) methodology. Due to its highly descriptive nature and model-centric approach [5], MBSE is an appropriate methodology to model systems on system and item level, making it useful in the development process of test cases for ML applications [6]. The concepts presented in the work at hand are, amongst others, exemplified by methods from MBSE.

This paper discusses the generation of test scenarios for an AI-based system. The use case is applying a computer vision algorithm to perform object detection and predict dangerous situations. The scenarios represent different situations with foreign airplanes used for testing. The detailed use case is explained in [7]. A method for iterative scenario-based testing of AI-based systems is presented in the scope of this work. Three essential parts of the method are defined: A high-level description of the scenarios to be executed, the testing environment in which test scenarios are executed, and a monitoring tool for defining the parameter boundaries for the ODD of the respective system. A prototypical implementation of this methodology is also presented. For modeling the systems involved and developing test cases, the MBSE tool Cameo¹ is used.

The simulation is executed in FlightGear, a highly customizable open-source software for flight simulation². The scenarios are generated in a model-based approach in Cameo and then executed in a FlightGear instance. Parameters are monitored using a custom Python library.

¹Dassault Systemes, 2022, Cameo Systems Modeler, available at <https://www.3ds.com/products-services/catia/products/no-magic/cameo-systems-modeler/>.

²FlightGear developers & contributors, 2021. FlightGear, Available at <https://www.flightgear.org/>.

The findings show that the iterative scenario-based testing method facilitates the definition and refinement of test scenarios for AI-based applications.

The remaining paper is structured as follows: In Section 1, related work and the status quo of scenario-based testing with a model-based approach are discussed. Section 2 presents the development of a domain-independent method for iterative scenario-based testing. The implementation of this methodology is presented in Section 3 with tools used for defining scenarios, executing them, and monitoring them.

1 Related Work

In [8], Jafer and Durak discuss the complexity of simulation scenario development in aviation. They propose ontology-based approaches to develop an aviation scenario definition language (ASDL). According to the authors, ontologies provide invaluable possibilities to tackle the complexity of simulation scenario development. Durak presents a model-driven engineering perspective for scenario development in [9]. The use of metamodels for generating executable scenarios is demonstrated with a sample implementation. Durak's work is closely related to the research presented in the work at hand, specifically the development of metamodels for generating executable scenarios.

Simulation-based data and scenario generation for AI-based airborne systems is discussed by Gupta in [10]. In the work, the authors aim to answer the question of what needs to be simulated for synthetic data and scenario generation in the simulation engineering process of an AI-based system. The used methods are a simulation-based data generation process adapted from EASA's first usable guidance for Level 1 machine learning applications and the scenario-based approach using SES, which is explained more thoroughly in the publications of Durak [11], [12] as well as Karmokar [13]. The work in [10] is succeeded with [14], which discusses behavioral modeling for scenario-based testing in aviation and introduces an enhanced approach for scenario-based testing called Operational Domain Driven Testing.

Closely related, [15] demonstrates the testing of black box systems, such as AI-based applications for autonomous road vehicles, in their ODD. The framework introduced by the authors is used to learn monitors in a feature space and prevent the system from using critical components when exiting its ODD.

Scenario-based testing of autonomous road vehicles is discussed in [16] and [17]. The authors present an automated scenario-based testing methodology for vehicles using advanced AI-based applications. The work shows that the presented formal simulation approach effectively finds relevant tests for track testing with a real autonomous vehicle.

In [18], Hungar introduces scenario-based testing for automated road vehicles. The outcome of later iterations [19] is the PEGASUS method, which is used to assess highly automated driving functions. According to the author, the most important steps for scenario-based testing involve capturing all evolutions, i.e. variants, of functional scenarios, formalization of them, systematic testing, the analysis of critical regions, and finally, the development of a risk chart.

Closely related to [9], the work presented in this paper discusses model-driven scenario development. In addition to the methodologies discussed in the related work, an iterative scenario parameter adjustment and generation process is introduced, forming the iterative scenario-based testing method. The method is illustrated with an exemplary generation of test scenarios for an AI-based demonstrator. In the next section, the methodology for this domain- and tool-independent iterative scenario-based testing method is presented.

2 Iterative Scenario-Based Testing Concept

The related work shows that there are many ways to realize scenario-based testing for AI-based systems. Especially when talking about domain-specific tools, a variety of testing strategies are possible. A generalization of these testing strategies can help with defining universal testing methods. To achieve that, a fundamental, tool-independent method is needed to describe the basic methodology for iterative scenario-based testing on a high level of abstraction. This method can then be used to build some domain-specific testing tools. For such iterative scenario-based testing, three fundamental components have been identified:

Scenario Model

First, a high-level description of the testing scenarios needs to be defined. This high-level model can be achieved by describing the scenarios' fundamental components. Modeling tools or formalized methods can

for instance be used to formulate the scenarios and derive all required scenario variations from the high-level model. The method shall be capable of generating an arbitrary number of scenarios with high parameter variation from the high-level description to achieve satisfactory test coverage for the application to be verified.

Environment for Scenario Execution

Second, an environment for executing the derived scenarios should be selected. The environment can be of different types, such as simulated, real system, or a mix of both, e.g. real systems extended with elements from augmented reality. These environments have different advantages and disadvantages. A simulated system can be deployed quickly, offers consistent conditions, and is cost-effective. The biggest drawback of simulated environments is their sim-to-real gap. The gap refers to the applicability of simulations to real-life environments, as many simulated environments cannot fully offer all relevant conditions as a real system. The biggest advantage of a real system is its smaller domain gap to the real-life environment in which the tested application is designed to operate in. Real systems are hard to deploy and costly. Especially when talking about automated and accelerated testing, real systems can pose a financial and temporal bottleneck in the testing process.

ODD Monitoring

Last, a monitoring tool is required for verification and for tracking all parameters that are necessary for and can have some variance on scenarios. By tracking these parameters and verifying the application to be tested, a precise ODD can be defined for the system. With feedback from the monitoring tool, parameters can be adjusted, or new values for the parameters can be chosen for a new iteration of scenario generation. The tools qualifying for monitoring, in the chain of scenario-based testing, are arbitrary. They merely need to be capable of monitoring parameters in real-time for synchronization purposes. The described method is of an iterative nature. Each component feeds the next with some information. This loop is depicted in Figure 1.

The execution of test scenarios can be accomplished in a simulated environment as well as with a real system. Although both approaches are important to consider, the method depicted in Figure 1 is tailored towards testing in simulated environments. For generating application-readable scenario descriptions with the scenario modeling tool, some application, e.g. script,

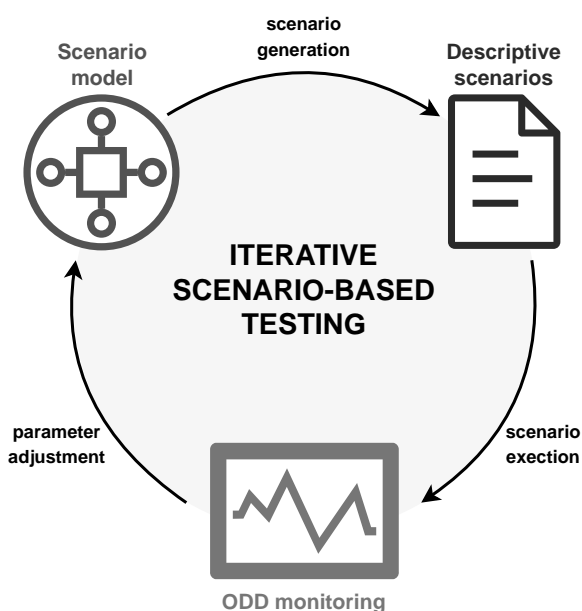


Figure 1: Iterative scenario-based testing

is needed. Similarly, after scenario execution and monitoring, some application is needed that feeds the result logs to the scenario modeling tool, decides on parameter adjustment, and triggers new scenario generation. The use of such intermediate applications and scripts enables high automation and optimization of the method. In ideal circumstances, the iterative scenario-based testing method forms a closed loop with automated test scenario generation, execution, and real-time monitoring of parameters.

3 Exemplary Implementation

This section explains an exemplary implementation to demonstrate the derived method. For the implementation, domain-specific tools were selected that can be exchanged depending on the use case. The exemplary implementation of the discussed method can be divided into three components: First, the MBSE-based scenario description and generation using Cameo; second, the execution of scenarios defined in generated XML files with the flight simulator FlightGear; and last, the monitoring of parameters during scenario execution with a custom ODD monitoring tool. The basic flow of information and the steps are illustrated in Figure 2.

The high-level model of the scenarios is described with a profile diagram in Cameo. Profile diagrams are

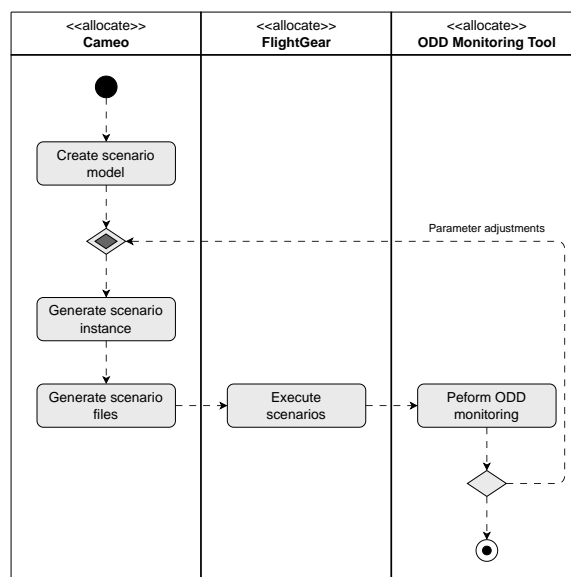


Figure 2: Flow of information and steps for iterative scenario-based testing used in this work.

defined in the System Modeling Language (SysML). Additionally, extensions are used to increase the modeling capabilities with profile diagrams.

One configuration of a specific scenario is generated with a block definition diagram, which can be transformed and exported into the desired XML scenario files with the help of scripts. XML files are generated for the use case on hand since FlightGear uses an XML format for the scenario definition. However, other domain-specific formats can be used as well. The scenarios are executed within an instance of FlightGear. A more detailed description of the implementation is provided in the following subsections.

3.1 Scenario Format and MBSE-Based Scenario Generation

A high-level description of the necessary files for scenario execution is displayed in Figure 3. Along with scenario files, flight plan files are needed for scenario execution.

The scenario files include various tags which define the inputs, objects, and attributes when executing them in FlightGear. An important tag is the <entry> tag which defines objects used in a scenario and can include the following additional tags: <callsign> – the identification of the airplane, <type> and <model>, <flight-plan> – the flight plan which the scenario refers to,

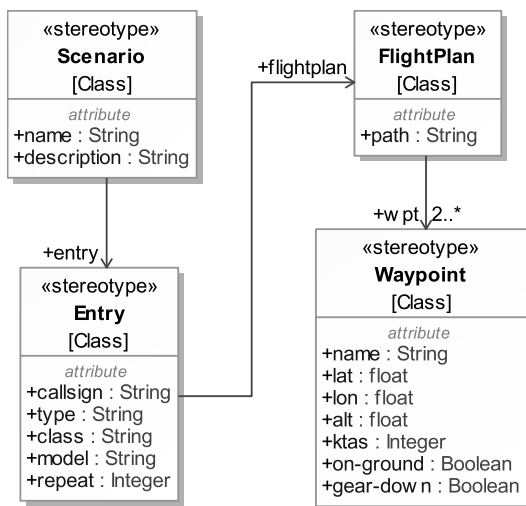


Figure 3: High-level description of the configuration files for FlightGear.

class – the class of airplane, and *<repeat>* – a Boolean flag that indicates whether the scenario is executed once or repeated infinitely often.

The flight plans, which the scenario files refer to, are also in XML format. The flight plan contains the *<wpt>* tag, which can include the following additional tags: *<name>* – the name of the waypoint, *<lat>* – the latitude of the entry that refers to the flight plan, *<lon>* – the longitude, *<alt>* – the altitude, *<ktas>* – the *knots true airspeed*, *<on-ground>* – if the specified object starts from the ground or not, *<gear-down>* – if the landing gear is retracted or extended, and *<flaps-down>* – for retracting or extending the flaps. FlightGear offers many more configuration files which can be adjusted to change environmental parameters as well as parameters of entities and other components of interest for scenario-based testing. For simplicity, only the scenario and flight plan files along with their parameters are discussed here. Some high-level description, i.e. metamodel, of the scenario and flight plan files is needed to generate arbitrary test scenarios.

Figure 4 depicts one instance of the high-level description of the scenario and flight plan files.

3.2 Scenario Execution

The scenarios are executed within FlightGear. The respective scenarios, in their descriptive XML format, can be executed manually in a FlightGear instance, or referred to as parameters for automatic execution with

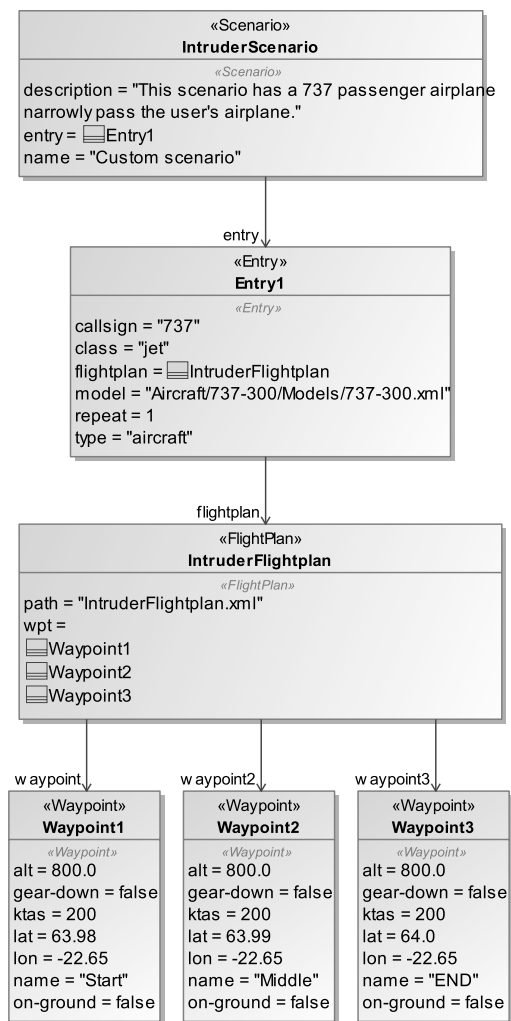


Figure 4: Block definition diagram of one scenario and flight plan configuration.

startup of FlightGear. For automation purposes, we chose the latter. As explained in the previous subsection, one or more entries, e.g. planes, can be defined in a scenario file, with each flying according to a predefined route.

In this instance, one passenger airplane is defined, which narrowly passes the user's plane. Both planes are flying towards each other. Figure 5 shows a screenshot of the scenario during execution in FlightGear.

3.3 ODD Definition and Monitoring

Several parameters can have a variance on the scenarios executed in FlightGear, some of which were defined above.



Figure 5: Passenger airplane narrowly passing user’s Cessna.

For a complete ODD definition, additional parameters, apart from the ones presented above, need to be considered. One example is environmental parameters such as weather conditions. The more operational parameters are considered within the ODD, the higher the test coverage of the scenarios and the narrower the sim-to-real gap. Naturally, parameters should be chosen based on their impact on the scenarios, and therefore the ODD. A high-level description of the domain model for the ODD of the AI-based system used on an airplane is depicted in Figure 6.

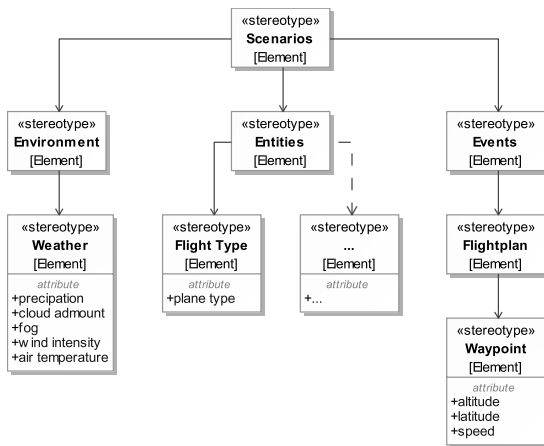


Figure 6: Domain model for the ODD of the scenario-based testing method.

The parameter boundaries for the use case of object detection during scenario execution can be determined in an iterative process. Due to the high number of parameters to be considered, a manual exhaustive search for parameter boundaries is highly time-consuming.

Therefore, some tool is needed which can track the necessary parameters during scenario execution and give feedback on the result of the tested application.

For monitoring these parameters in FlightGear, a public Python library³ for fetching parameters from FlightGear’s property tree is used. The AI-based system tested in this example is an object detection application. The result of the object detection, within the operational domain, and desired parameters can be logged using the monitoring tool. The feedback generated from the tool can then be used to adjust the values of selected parameters in Cameo according to the ODD boundaries, generate new scenarios, and redefine the ODD of the application. This iterative process can be used to define the ODD boundaries of each parameter with every iteration.

A simplified ODD for the system with the two parameters altitude and speed can be defined as follows: “The application performs correct object detection of intruding airplanes of type Boeing 737 within following parameter boundaries:

- Altitude of intruding airplane relative to own airplane in feet, Δ alt: -100 to 100.
- Cumulative speed of intruding airplane as well as own airplane in knots true airspeed, Σ ktas: 0 to 500.”

Table 1 shows an exemplary log recorded during execution of a scenario in FlightGear.

Log #	lat	lon	Δ alt	Σ ktas	detect
1	63.970	-22.65	100	400	no
2	63.974	-22.65	100	399	no
3	63.978	-22.65	100	400	no
4	63.982	-22.65	100	399	yes
5	63.986	-22.65	100	400	yes
6	63.990	-22.65	100	403	yes
7	63.994	-22.65	100	399	no
8	63.998	-22.65	100	400	no

Table 1: Exemplary log of parameters monitored during scenario execution in FlightGear.

For completeness and to reflect other relevant parameters currently covered in the scenario model, the

³Munyakabera Jean Claude, 2022. flightgear_interface, Available at: https://github.com/ironmann250/flightgear_interface

latitude and longitude of the intruding airplane are logged along with the aforementioned altitude and speed.

The table shows that successful object detection is on hand for logs four to six. Therefore, the predefined ODD holds for the combination of parameters on hand. Now, single parameters can be adjusted for a potential reevaluation of the predefined ODD. In this case, the altitude of the intruding airplane relative to the own airplane is increased by 100 feet. First, a scenario with a new configuration of attributes needs to be generated, similar to the one depicted in Figure 4. In this case, the altitude is adjusted to reflect the definition for the new test case. Lastly, the necessary XML files are generated from the configuration model.

Now, scenario execution in FlightGear and parameter monitoring can be performed. Table 2 shows the log for the second iteration of parameter monitoring.

Log #	lat	lon	Δ alt	Σ ktas	detect
1	63.970	-22.65	200	400	no
2	63.974	-22.65	200	399	no
3	63.978	-22.65	200	401	yes
4	63.982	-22.65	200	400	yes
5	63.986	-22.65	200	400	yes
6	63.990	-22.65	200	400	no
7	63.994	-22.65	200	400	no
8	63.998	-22.65	200	400	no

Table 2: Exemplary log of parameters monitored during scenario execution in FlightGear.

As shown in the second table, the object detection is successful for logs three to five. The predefined ODD still holds for the combination of parameters on hand. However, the ODD can now be adjusted and phrased more precisely in line with the altered parameter. The ODD for the application can therefore be rephrased as follows:

“The application performs correct object detection of intruding airplanes of type Boeing 737 within following parameter boundaries:

- Altitude of intruding airplane relative to own airplane in feet, Δ alt: -100 to 200.
- [...]”

The loop of parameter adjustment, scenario generation, execution, and monitoring can be repeated until the changes in detection performance drop below some predefined threshold and a sufficiently precise ODD has been determined. Note that precision does not refer to some performance measure, but rather to the perceived precision which is currently chosen arbitrarily.

The example for ODD monitoring and parameter adjustment presented above is simplistic and, for instance, does not consider constraints. Many more parameters can be and need to be considered when defining a precise ODD for the underlying application. Also, the granularity for testing parameter boundaries of the ODD needs to be determined accurately. For instance, a higher logging frequency of parameters can be chosen, which offers a more continuous data stream but also increases the analysis effort. Also, the more granular confidence of the object detector from the machine learning application can be used as a performance metric in the detection definition. The framework in itself requires fine-tuning and more testing to provide the right conditions for successful iterative scenario-based testing of various systems.

The exemplary implementation of model-based scenario generation and ODD monitoring in this section follows the method presented in Figure 1. Domain-specific tools utilizing Cameo, XML files, and a Python application were used to build a framework for iterative scenario-based testing.

The implementation can be seen as a minimal working example, demonstrating the iterative scenario-based testing method explained in Section 2. The implementation can be developed further to allow for closed-loop scenario-based testing with automated scenario generation, execution, and monitoring.

4 Conclusion and Discussion

The use of ML applications in systems such as airplanes is steadily increasing. The thorough testing of these systems is a fundamental part of their development process. Certain industries, such as aviation, impose strict requirements and constraints for the use of AI-based applications, increasing the testing efforts required to certify and use these applications.

Additionally, ML applications are often considered a black box. Therefore, black box testing methods need to be put in place that are as rigorous as current testing methods for common software systems.

This work depicts a method for iteratively testing an AI-based system that performs object detection in an airplane. For this purpose, a scenario-based testing loop was developed, including the three steps of generating application-readable scenario descriptions from models, execution of these scenarios, and parameter monitoring with model parameter adjustments.

In addition to generating arbitrary test cases, the presented method illustrates the approximation of boundaries for the ODD of the ML application with iterative parameter adjustments.

This method can be further optimized by connecting its components, i.e. the high-level scenario description, scenario execution, and ODD monitoring, and creating a closed loop with automated scenario generation, execution, and parameter adjustment.

Additionally, test oracles that determine the success or failure of individual tests should be investigated. The granularity of test cases, i.e. only success and failure evaluations or more finely-grained evaluations, is important. These findings will be investigated in future research.

References

- [1] EASA. *EASA Artificial Intelligence concept paper - proposed Issue 2*. Tech. rep. Feb. 2023. URL: <https://www.easa.europa.eu/en/downloads/137631/en>.
- [2] EUROCAE WG-114/SAE and G-34 Artificial Intelligence Working Group. *Artificial Intelligence in Aeronautical Systems: Statement of Concerns*. SAE International. Apr. 2021. DOI: <https://doi.org/10.4271/AIR6988>.
- [3] On-Road Automated Driving (ORAD) Committee. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. Standard. SAE International, Apr. 2021. DOI: https://doi.org/10.4271/J3016_202104.
- [4] *DO-178C - Software Considerations in Airborne Systems and Equipment Certification*. Standard. RTCA, Dec. 2011. URL: <https://www.d0178.org/>.
- [5] A. Wayne Wymore. *Model-Based Systems Engineering*. 1993. ISBN: 9780203746936. DOI: 10.1201/9780203746936.
- [6] Azad M. Madni. "MBSE Testbed for Rapid, Cost-Effective Prototyping and Evaluation of System Modeling Approaches". In: *Applied Sciences* 11.5 (2021). ISSN: 2076-3417. DOI: 10.3390/app11052321.
- [7] Jasper Sprockhoff et al. "Model-Based Systems Engineering for AI-Based Systems". In: *AIAA SCITECH 2023 Forum*. Jan. 2023. DOI: 10.2514/6.2023-2587.
- [8] Shafagh Jafer and Umut Durak. "Tackling the Complexity of Simulation Scenario Development in Aviation". In: *Proceedings of the Symposium on Modeling and Simulation of Complexity in Intelligent, Adaptive and Autonomous Systems*. 2017. ISBN: 9781510840300.
- [9] Umut Durak et al. "Scenario Development: A Model-Driven Engineering Perspective". In: *4th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*. 2014, pp. 117–124. URL: <https://elib.dlr.de/94626/>.
- [10] Siddhartha Gupta et al. "From Operational Scenarios to Synthetic Data: Simulation-Based Data Generation for AI-Based Airborne Systems". In: *AIAA SCITECH Forum*. Jan. 2022. DOI: 10.2514/6.2022-2103.
- [11] Umut Durak et al. "Using System Entity Structures to Model the Elements of a Scenario in a Research Flight Simulator". In: *AIAA Modeling and Simulation Technologies Conference*. 2017. URL: <https://elib.dlr.de/112664/>.
- [12] Umut Durak et al. "Computational Representation for a Simulation Scenario Definition Language". In: *Modeling and Simulation Technologies Conference*. 2018. DOI: 10.2514/6.2018-1398.
- [13] Bikash Chandra Karmokar et al. "Tools for Scenario Development Using System Entity Structures". In: *AIAA Scitech Forum, 2019*. 2019. DOI: 10.2514/6.2019-1712.
- [14] Siddhartha Gupta and Umut Durak. "Behavioural Modeling for Scenario-based Testing in Aviation". In: *AIAA SCITECH Forum (not yet published)*. Jan. 2023.
- [15] Hazem Torfah et al. "Learning Monitorable Operational Design Domains for Assured Autonomy". In: *Proceedings of the International Symposium on Automated Technology for Verification and Analysis (ATVA)*. Oct. 2022.
- [16] Daniel J. Fremont et al. "Formal Scenario-Based Testing of Autonomous Vehicles: From Simulation to the Real World". In: *IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)* (2020), pp. 1–8. URL: <https://api.semanticscholar.org/CorpusID:212736906>.
- [17] Francis Indaheng et al. "A Scenario-Based Platform for Testing Autonomous Vehicle Behavior Prediction Models in Simulation". In: *ArXiv abs/2110.14870* (2021).
- [18] Hardi Hungar. "A Concept of Scenario Space Exploration with Criticality Coverage Guarantees -Extended Abstract". In: *9th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation, ISoLA 2020*. 2020, pp. 293–306. URL: <https://elib.dlr.de/137353/>.
- [19] Hardi Hungar. "Scenario space exploration for establishing the safety of automated vehicles". In: *3rd China Autonomous Driving Testing Technology Innovation Conference*. Dec. 2020. URL: <https://elib.dlr.de/139626/>.

A Use Case for Digital Tools in Crafts: Simulation and Virtual Reality for Carpentries

Bastian Prell*, Jörg Reiff-Stephan

Technical University of Applied Sciences Wildau, Hochschulring 1, 15745 Wildau, Germany

*bastian.prell@th-wildau.de

SNE 33(4), 2023, 191-198, DOI: 10.11128/sne.33.tn.10667
Selected ASIM WS 2023 Postconf. Publication: 2023-10-15
Rec. Rev. Impr. & Modified: 2023-11-30; Accepted: 2023-12-06
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. This paper presents an integrated approach to explore human factors in the craft sector. The embedded use case consists of craft workshops being 3D-modelled and simulated so that different scheduling rules can be deployed. The connected VR visualization allows craft professionals to experience scheduling rules and their impact on different performance measurements in an environment intuitively resembling their actual workshops. The lessons learned from the simulation modelling are collected and presented. A modular approach appears to be beneficial for simulation modelling, offering flexibility and pragmatism but detail when needed. Apart from this anecdotal evidence, the study is accompanied by interviews. These are analyzed based on abduction grounded theory which combines a strong focus on the actual corpus of qualitative data but considers theoretical foundations, when appropriate. This paper presents the study design, and how it interties with the research approach and the use case of simulation and virtual reality for carpentries.

Introduction

Industry 4.0 is a well-known term for production scientists and practitioners. However, the typical craft workshop is usually only partially, if at all, automated. While the selective use of automation can be justified by smaller production quantities and lower economic efficiency, such argumentation cannot fully transfer to digitisation trends. Industry 4.0 solutions often had a strong technical focus. In contrast to that, most recent approaches, with the label Industry 5.0 pay more attention to human factors in the production environment [1, 2]. This may offer similarity to the value creation in the craft sector.

One tool used in this context is discrete event simulation (DES) to model and simulate material flows through the production facility. DES is linked to the term digital factory. For example, it is used to virtually assess production layouts for production facilities both new and re-planned or different options of production scheduling. In contrast to that, the craft sector lags in applying digital tools. However, the requirements from both areas show certain similarities, as both value on-time delivery, throughput, capacity utilisation, lead times, etc.

1 Research Context

Compared to industrial value creation, increased labour productivity has only been accomplished to a lesser extent in the craft sector. Among other things, crafts are characterised by the fact that tools are operated manually. In contrast to that, industrial processes are mostly characterised by machine-guided tools. Ancillary processes, however, are not covered by this distinction but are similar to those from the industrial sector. Productivity describes the relationship between input factors and output over time, which is closely linked to the term technology from an economics point of view. New technologies are usually described by the term innovation. This section presents a brief introduction to current state of the art for craft research, innovation diffusion, technology acceptance, simulation, and virtual reality (VR).

1.1 Craft Research

The craft sector is multifaceted and almost exclusively characterised by mostly small and some medium-sized enterprises [3]. Crafts account for 12.4 % of employment and 25 % of apprenticeships in Germany. Roughly 8 % of Germany's gross domestic product (GDP) is generated in the craft sector [4]. With the disproportionately large share of apprentices, it can be assumed on the one hand that there is sufficient digital literacy [5]; on the other hand, that the integration of digital content into training is of even greater significance [6].

Compared to enterprises from the industrial sector, digital tools are used less intensively in the craft sector [7]. The difference and the need to catch up in terms of digitalisation, have already been the subject of various studies [8–11].

The craft sector has been affected disproportionately by the demographic change and will continue to do so: A large proportion of workers in this sector will retire in the next five to ten years. The general demographic of smaller cohorts following the so-called baby boomers won't be able to compensate for that. Furthermore, it can be observed that the choice of training by these overall fewer potential applicants, also in relative terms, was taken less and less in favour of craft professions [12].

At the same time, it is assumed that the demand for craft professionals will remain constant or even increase [13]. Thus, everything points to an increasing labour shortage in this sector. This calls for new technologies that could potentially increase labour productivity. Thus, innovations need to emerge or be transferred and spread (diffuse) throughout the craft sector.

1.2 Innovation Diffusion

The fact that the craft sector does not use the potential of existing technologies to the same extent as the industrial sector raises the question of how innovations emerge in the craft sector. Innovation theory has produced various models to describe innovation processes.

The traditional linear innovation model [14], which is based on a neo-classical understanding and thus often assumes an exogenous technology push, does not seem to be able to explain the different degrees of utilisation in industry and craft: Technology and utilisation are open to both sectors.

Considering individual perceptions and mutual influence between individuals and sectors seems a better fit for the focused craft sector. More current models from the group of systemic approaches, such as the *Technological Innovations Systems Approach*, for example, include those behavioural aspects [15].

Such behavioural aspects can arise, for example, from individual acceptance (or non-acceptance in case of rejection). Acceptance has been formalised through different models, whereas the *Technology Acceptance Model* (TAM) is commonly used. TAM has emerged less systematically than the aforementioned innovation theory models, but from requirements management for specific solutions [16].

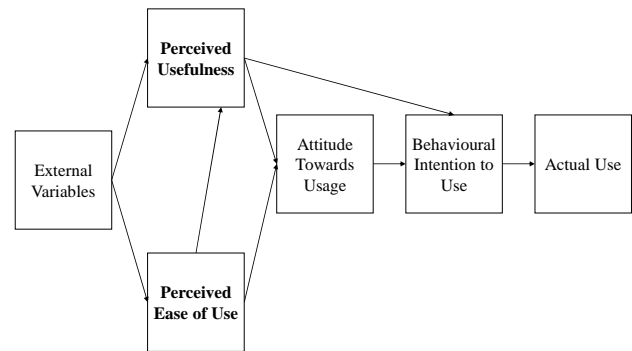


Figure 1. Modelled Technology Acceptance.

Originally designed to assess the acceptance of information and communication technology, especially computer programs, various extensions (TAM2 and TAM3) have been established for a wide range of applications and have been proven to be robust [17]. The TAM assumes that acceptance is directly related to the actual use of a certain technology [18], as it is depicted in Figure 1.

In the context of basic innovation theory terminology, rooted in Schumpeter's work, this relates to what makes up an innovation in the first place: Namely, an invention or idea that becomes established by wide application [19]. The TAM supports the interviewing of probands pre and post exposition to novel technologies to determine influencing factors.

1.3 Simulation in Production and Logistics

“A simulation is the imitation of the operation of a real-world process or system over time” which may be advised for a broad range of applications from operations research and systems analysis [20]. Simulation has been established for both planning new and replanning existing processes, systems, or factories in production and logistics. While a variety of different software suites has evolved to cover a wide area of application, the method of DES has become especially popular for many scopes in production and logistics.

Bracht et al. classified visualisation as either to be dynamic or static to describe whether the visualisation is dynamically changing during the simulation run [21, 22]. Animation is a special type of visualisation, according to VDI 3633 Sheet 11 [23]. Nevertheless, animation is no synonym for simulation, which describes the method of computing a system's behaviour [22]. Wenzel points out that not only insights but also the communication of these insights ought to be achieved by simulation [24]. Common use cases are found in the automotive sector, commissioning, chemical industry, or food industry.

It is striking that non-industrial production has not been examined with simulations, even though layout planning or scheduling is relevant in this sector as well. Recent developments have led to the establishment of 3D-visualisation, which is easily understood also by non-experts.

1.4 Virtual Reality

Digitisation and process optimisation projects are often accompanied by the expectation that the results will have a positive impact on the profitability of value creation and are always associated with a risk of failure due to project-inherent uncertainties [25].

Hence, early transparency is of great importance, which can be supported by VR. Figure 2 depicts the so-called reality-virtuality continuum, according to [26]: The most left pole represents the real environment, with no virtuality. The augmentation of real objects by context-based information is called augmented reality. The reason for that is, that real objects make up most elements in the user's field of vision resp. perception [27].

Meanwhile, VR is usually created by displays that exclude real-world objects. Instead, a virtually created world is displayed and mostly synchronised to head movements to realistically stimulate human senses [28]. VR is not enabling a completely virtual environment, as humans, generally speaking, have further senses that are still connected to reality and not yet stimulated virtually, e.g. sense of balance, smelling, etc. Together, AR and VR make up the term *mixed reality* [29].

According to the classification of Reif either head-mounted displays or emulators support the visualisation. For the proposed use case, the HTC Vive counts as a head-mounted display, which is powered by an external computer, running the simulation [30]. The provided hardware allows six degrees of freedom, for best immersion: Three for rotative movements, usually of the head like yawing, rolling and pitching, and the three translatory motion forward, leftward, or rightward [31].

VR applications can be already found in practice: Education, production planning, or process optimisation as well as the visualisation of products for customers are common applications [32].

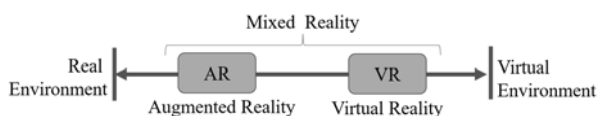


Figure 2. Reality-Virtuality Continuum according to [26].

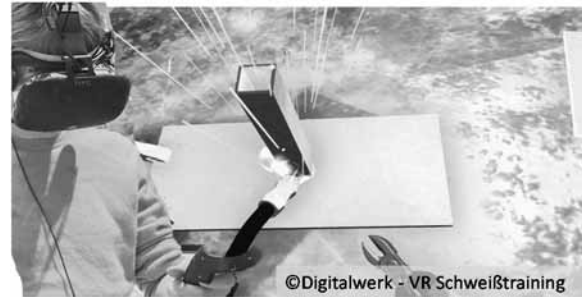


Figure 3. Welding Training in VR.

One study found training, planning, and communication to be the most relevant topics for VR in crafts [33]. Figure 3 shows a welding simulation, which is one of the examples used in the aforementioned study. Furthermore, visualisation is stated for planning, even though the authors don't state layout planning explicitly for production environments but kitchens.

VR offers a rather intuitive orientation and the possibility of a controlled environment that can be of high value, e.g. in training scenarios that are not feasible otherwise, as they might be too expensive or too dangerous. A potential drawback of such a technology is the so-called motion sickness [34].

2 Research Design

The research design employs use case analysis as it is common in engineering science to anecdotally pass on experiences of designing and implementing technical systems. That way, the lessons learned can be deduced and documented. The novelty of this topic strongly points towards explorative research. That is why the proposed design integrates grounded theory from the field of social science. It is used to create hypotheses about human behaviour and social life from field data, which consists mainly of interview transcriptions and observations [35].

More precisely, abduction grounded theory is used, as theoretical elements from literature, as presented in Section 1, are considered as well. The use case presents a convenient way to get in touch with the research object (craft professionals) and expose them to DES and scheduling, which are state-of-the-art industrial and digital methods. This *technology shock* is then used to track reactions from individuals and their associated groups, respectively the company. The remainder of this paper discusses the identified use case and the planned data collection and analysis, while further aspects of the research design are displayed in Section 4, once the use case has been described.

3 Use Case Description and Realisation

This section describes the activities planned to create a DES model and to display scheduling alternatives to craft professionals. Figure 4 represents the overall model of procedure. The main subtasks were adopted to a reduced extent from VDI 3633 [36]. The standard process (grey arrow) is deployed on an abstract level. This is to create modules of different layouts that may occur or make up actual workshops of craft companies. These modules were defined as flow shop or job shop layouts.

Hence, modular simulation models for each of these two are implemented (blue arrow) using synthesised data, representing typical workshops of this sector. For the modelling of the actual workshop, the real-world data needs to replace the modules' dummy data, and parameters need to be set accordingly. Especially the 3D visualisation needs to reflect the specifics of the craft workshop. This can be done by using 3D-design software such as Blender or CAD software like SolidWorks or Autodesk Inventor.

These 3D objects are then imported into the simulation software by Siemens Tecnomatix Plant Simulation. Only machines relevant for the schedule are designed in detail to ensure reasonable usage of resources. Other machinery or the workshop surrounding can be integrated as panes displaying 2D-photos of the respective objects. This allows a shortcut in modelling while ensuring intuitive resemblance. Finally, a functioning virtual 3D image of the workshop is created (orange arrow).

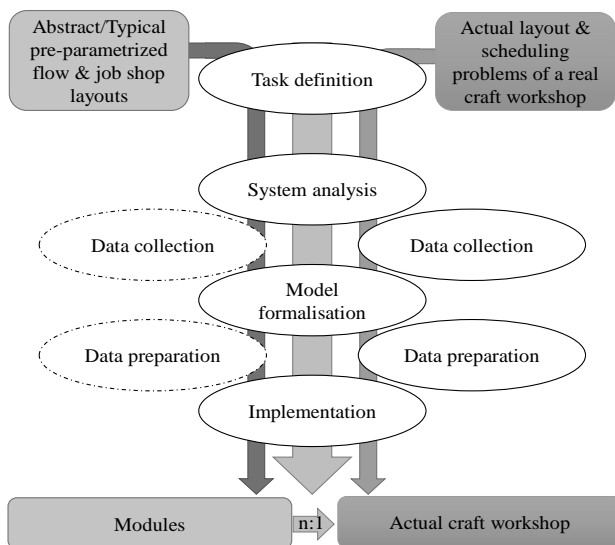


Figure 4. Model of Procedure for Simulation Creation.

3.1 Layout Patterns

This subsection presents the two main layout options that are thought to modular build up most craft workshops: Flow shop and job shop, which also determine the respective scheduling problems.

Flow Shop Problem (FSP) represents a scheduling problem where all jobs have the same processing sequence [37]. Usually, a flow shop is characterised by a flow-oriented layout, so that machines are arranged in series, as shown in Figure 5. This allows jobs to flow from an initial machine, through several intermediate machines to the final machine. This means that each operation after the first has exactly one direct predecessor, and each operation except the final machine has exactly one direct successor [38].

Common objectives for FSPs are to find schedules that minimise longest completion time or makespan while maximising machine utilisation. This possibly leads to the highest throughput [39]. Flow shop production is usually associated with portfolios of only a few, highly standardised products, with predictable demand. Thus, the most common production strategy is *make-to-stock*. The number of constraints to consider, when scheduling, is significantly less than for a job shop layout. For instance, the setup time is rarely considered since setups do not occur often. A flow shop provides good visibility of occurring problems due to the linear layout.

Job Shop Problem (JSP) consists of jobs, running through a common production facility but having different machine orders [40]. Jilcha & Berhan [41] and Li et al. [42] present JSP using the following definition: Job shop scheduling happens in a work location of a given set of general-purpose workstations where a variety of jobs, each with a specific set of operations, is processed in a given sequence.

Jobs, therefore, have their individual routing and might use all machines of the respective layout or a subset of these (see Figure 5).

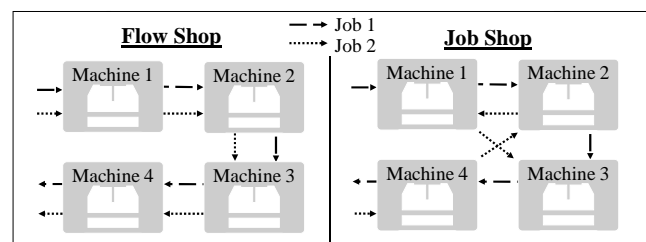


Figure 5. Flow Shop and Job Shop Layout Principles.

A JSP's objective usually is to minimise the makespan, subject to the constraints specified for each job. Overall characteristics of production in a job shop environment can usually be summarised as follows: Firstly, the variety of products is very high due to the possibility of customisation.

Therefore, the production layout requires high flexibility which leads to low process standardisation. Additionally, demand is harder to predict than for the typical FSP-associated product portfolio since it usually operates under a make-to-order strategy. Admittedly, not only customised, or single pieces are produced in job shop configuration, but small batches as well. Finally, regarding the workers' skills, professionals working in job shop layouts need to master more tasks. Hence, they require more training in their respective qualifications.

Scheduling production in a job shop environment should consider the following aspects: The make-to-order strategy is linked to the target of on-time delivery. Overlooking the process is much more complex since every product has its specific routing.

3.2 Priority Rule-based Scheduling

Priority rules have been used for decades as a scheduling procedure for production. Its main objective is to define the sequence in which the pending jobs are processed, in a given period. It is known that priority rules will usually provide inferior solutions in comparison to what may be given by other advanced algorithms, e.g., genetic algorithms, neural networks, simulated annealing, etc. However, it is commonly used because its implementation is easy and does not require expert knowledge nor skills [43]. As more complex approaches may be feasible for some craft companies, the authors assess rule-based scheduling to be more suitable for most companies in the craft sector. Hence, scheduling shall be limited in this paper's scope to priority rules. The Encyclopedia of Production and Manufacturing Management [44], as well as Koruca & Aydemir [43], present in their work the most commonly used rules for production scheduling. Ten of these rules are introduced in Table 1 and will be implemented in the simulation modules.

Figure 6 depicts the implications from these rules, using 5 jobs (J1-J5) with varying process times.

Rule	Description
First come, First served (FCFS)	Jobs will be scheduled and processed according to which was first to arrive either in the queue or to the machine.
Last come, first served (LCFS)	The last job to arrive is the one that is scheduled or processed next.
Shortest processing time (SPT)	When scheduling the job, the one with the shortest processing time among those in the queue is processed next. This reduces the work in process inventory, the total flow time, and the average job lateness.
Longest processing time (LPT)	This rule will schedule the job with the longest processing time among the jobs in the queue, next. While using this rule for scheduling the total completion time or makespan will be minimised.
Earliest due date (EDD)	When applying this rule, the job with the earliest due date will be processed next. The main aim of the rule is to reduce job lateness.
Shortest remaining processing time (SRPT)	The job that will have the priority of being processed is the one that has the shortest remaining processing time. The main aim of this rule is to minimise the total completion time or makespan and minimise the latest job delivery time.
Longest remaining processing time (LRPT)	The job that has the longest remaining processing time will be processed next by the machine. The aim of this rule is to maximise the capacity utilisation of machines/workstations in the work location.
Slack time (ST)	Is a variant of earliest due date rule, considering the remaining processing time (setups and lead time). The job that has the smallest amount of slack gets top priority. This rule integrates customer orientation and capacity utilisation.
Next Queue (NQ)	While scheduling, the NQ rule considers the queues at each of the succeeding machines/workstations where the jobs are heading. The priority goes to the job whose machine/workstation has the smallest queue.

Table 1: Priority-based scheduling rules.

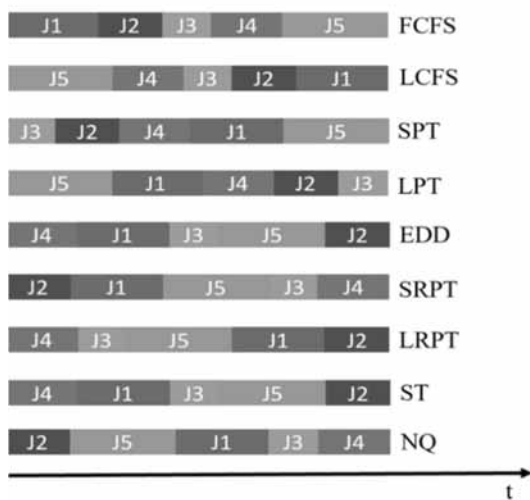


Figure 6. Priority-based scheduling rules, and their exemplary impact on the sequence of jobs

4 Further Course of the Study

According to the outlined research design, the use case of scheduling is embedded in the simulation and application of VR by craft professionals. The following subsections cover the methodological integration and the interviews.

4.1 Study and Use Case Integration

The use case analysis and the human factor analysis can be seen as two outcomes of this integrated work, which are different but not separate, as presented in Figure 7.

The work on the use case is currently ongoing as well as the acquisition of cooperating companies. The research questions have been defined as:

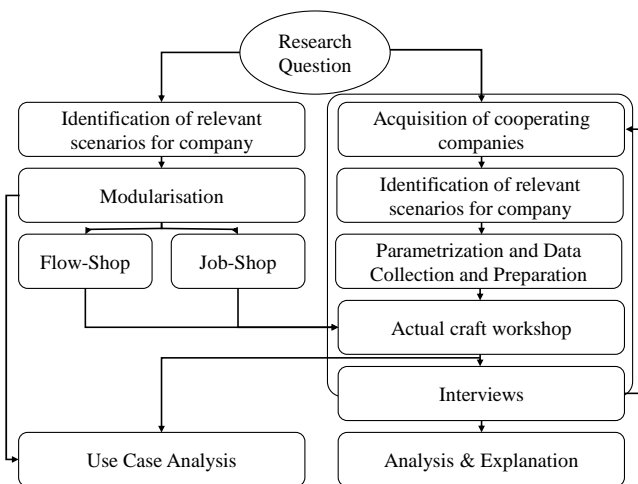


Figure 7. Model of Procedure for Conducting the Study .

- How is the innovation process characterised in the crafts sector? What roles do various individual and social stakeholders play?
- What kind of innovation-type classifications are feasible for craft companies?
- Which special features can promote the prevalence of novel technologies and how are they related?

The presented research design offers a path to find underpinning explanatory hypotheses for those research questions. The study that encloses the actual applications is designed and the data collection and analysis are prepared according to the following subsection. A minimal feasible model has been created and software evaluated that allows realistic but also pragmatic modelling of the actual machinery. The adaption to the first actual workshop is the next step to be taken.

4.2 Interview Preparation

Conversational interviews are the preferred interview technique for qualitative research at the exploratory stage [45]. Semi-structured guidance can promote story-telling by the interviewees. The target is to capture accounts of experience and behaviour related to the research question respectively leading to hypotheses. Thus, the theoretical background of Section 1 was considered for the design of the guiding questions [46]. Nonetheless, the interviews should only be carefully guided, avoiding any priming of interviewees when it comes to attitudes and sentiments.

Guidance can try to steer the conversation towards topics of internal or external communication, the used channels of communication and their recipients, usefulness, job relevance, ease of use, the perceived benefit and usage intention. For a similar approach, assessing a different use case the authors thought it useful to not reveal those subtopics directly nor the research question to achieve honest accounts. Therefore, the conversation was supported by a SWOT (Strength, Weakness, Opportunity and Weaknesses) template (Figure 8).

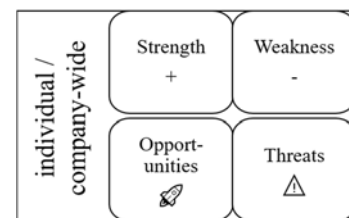


Figure 8. SWOT Framework for Interview Guidance.

This template is shortly explained to retain the interviewees' attention for the rest of the interview. This was assessed to cover the actual research questions well. Nevertheless, the interviewees ought to be made aware of the abstract topic of technology transfer into the craft sector for reasons of integrity. Provided the interviewees give their permission, the interviews are to be directly voice recorded or otherwise transcribed in great detail. Furthermore, impressions from the interview setting should be noted.

This so-called corpus of qualitative data is then analysed by interpretation, identification, and assignment of patterns and the deduction of explanatory hypotheses.

5 Conclusion and Outlook

This paper presents a mixed-methods approach to exploratively study the acceptance and diffusion of innovations within the craft sector while realising a DES use case. A DES model of an actual craft workshop is intended to employ different scheduling scenarios, and show the effect on different performance indicators that are relevant across the sectors of industry and crafts. Scheduling, in this case, represents an industrial method that is demonstrated using digital technology, which is novel to the craft sector.

Both, method and technology are not used extensively in crafts and, therefore, offer innovation potential. How craft professionals assess this potential and respective usage within their field of work on an individual but also a company-wide level is explored through accompanying conversational interviews.

This paper presents the outline of the research design and the underpinned model of procedure for the actual realisation of the use case. The study is currently at the phase of the use-case realisation and a minimal viable DES has been implemented. The authors hope to employ the proposed design on different occasions, receiving thick feedback data that can be documented and analysed, to formulate detailed hypotheses for technology acceptance and innovation diffusion in the craft sector.

Acknowledgements

This publication was made possible through the funding of the PhD program Innovation and Career Center - Integrated Engineering by the state of Brandenburg's Ministry of Research, Education and Culture MWFK (Germany).

References

- [1] Günther N, Reiff-Stephan J. Prescriptive Education im Zuge der Industrie 5.0. In: Jäkel, J. and Thiel, R. (eds.) AALE. pp. 315–320. VDE Verlag, Leipzig (2020). DOI 10.1007/978-3-642-51663-4_17
- [2] Prescher T, Hellriegel J, Schön M, Baumann A, Heil M, Schulz F. Digitalisierung im Handwerk als Lernprozess fördern. CEUR Workshop Proc. 1669, 209–215 (2016).
- [3] Statistisches Bundesamt: Produzierendes Gewerbe. (2019). DOI <https://doi.org/2040720177004>
- [4] Zentralverband des deutschen Handwerks: Kennzahlen des Handwerks, <https://www.zdh.de/daten-fakten/kennzahlen-des-handwerks/>, last accessed 2020/06/23.
- [5] Günther N, Reiff-Stephan J. Future Skills für die Produktion von Morgen. In: AALE 2019, (2019).
- [6] IG Metall: Das große Ding: Handwerk 4.0 Wie Betriebe und Beschäftigte erfolgreich den digitalen Wandel meistern Die Gewerkschaft für das Handwerk. (2015).
- [7] Peters T. Branchenanalyse SHK-Handwerk: Aktuelle Herausforderungen und Chancen. (2016).
- [8] Mittelstand 4.0-Kompetenzzentrum Hamburg: Digitalisierung im Hamburger Handwerk: Status Quo und Bedarfe 2021, <https://www.kompetenzzentrum-hamburg.digital/themen/handwerk/355-handwer-status-quo-und-bedarfe-2021>, last accessed 2021/12/03.
- [9] Owen A, Plöger W, Hiltner G, Reith A. Digitalisierungsbarometer für das Bau- und Ausbauhandwerk Kurzfassung Eine empirische 360° Analyse. (2020).
- [10] Thonipara A, Höhle D, Proeger T, Bizer K. Digitalisierung im Handwerk - ein Forschungsüberblick. Göttinger Beiträge zur Handw. 36, (2020). DOI 10.3249/2364-3897-gbh-36.
- [11] Veltkamp, N., Schulte, K.-S.: Digitalisierung des Handwerks. (2020).
- [12] Müller K. Dialog und Perspektive Handwerk 2025. I, (2016).
- [13] Creditreform Wirtschaftsforschung: Wirtschaftslage und Finanzierung im Mittelstand 2020/2021. 1–5 (2021).
- [14] Rogers EM. Diffusion of innovations. Free Press, New York, NY [u.a.] (2003).
- [15] Greenacre P, Gross R, Speirs J. Innovation Theory: A review of the literature. ICEPT Work. Pap. 11, 1–46 (2012).
- [16] Davis FD. User Acceptance of Information Systems: The Technology Acceptance Model (TAM), <http://www.jstor.org/stable/249008>, (1989).
- [17] Venkatesh V, Davis FD. Theoretical extension of the Technology Acceptance Model: Four longitudinal field studies. Manage. Sci. 46, 186–204 (2000). DOI 10.1287/mnsc.46.2.186.11926.
- [18] Jockisch M. Das Technologieakzeptanzmodell, (2010). DOI 10.1007/978-3-8349-8484-5_11.

- [19] Hauschildt J, Salomon S. Innovationsmanagement. Vahlen, München (2011).
- [20] Banks J, Carson J, Nelson B, Nicol D. Discrete-event System Simulation. Prentice Hall (2010).
- [21] Bracht U, Geckler D, Wenzel S. Digitale Fabrik. Springer Heidelberg Dordrecht London New York (2011). DOI 10.1007/978-3-540-88973-1.
- [22] Gutenschwager K, Rabe M, Spieckermann S, Wenzel S. Simulation in Produktion und Logistik. Springer Berlin Heidelberg, Berlin, Heidelberg (2017). DOI 10.1007/978-3-662-55745-7.
- [23] VDI Verein Deutscher Ingenieure: VDI Richtlinie 3633, Einführungsblatt (2014).
- [24] Wenzel S. Verbesserung der Informationsgestaltung in der Simulationstechnik unter Nutzung autonomer Visualisierungswerkzeuge. Verlag Praxiswissen, Dortmund (1998).
- [25] Knothe T, Reiff-Stephan J, Vladova G, Ullrich A. Industrie 4.0 im Produktionsumfeld. In: Metamorphose zur intelligenten und vernetzten Fabrik. Springer Vieweg, Berlin (2017). DOI 10.1007/978-3-662-54317-7.
- [26] Milgram P, Takemura H, Utsumi A, Kishino F. Augmented reality: A class of displays on the reality-virtuality continuum. In: Telemanipulator and Telepresence Technologies. pp. 282–292 (1995). DOI 10.1117/12.197321.
- [27] Azuma RT. A Survey of Augmented Reality. Presence Teleoperators Virtual Environ. 6, 355–385 (1997). DOI 10.1162/pres.1997.6.4.355.
- [28] Steuer J. Defining Virtual Reality: Dimensions Determining Telepresence. J. Commun. 42, 73–93 (1992). DOI 10.1111/j.1460-2466.1992.tb00812.x.
- [29] Grothus A, Thesing T, Feldmann C. Digitale Geschäftsmodell-Innovation mit Augmented Reality und Virtual Reality. Springer Berlin Heidelberg, (2021). DOI 10.1007/978-3-662-63746-3.
- [30] Reif R. Development and Evaluation of an Augmented Reality Order Picking System, (2009).
- [31] Prithul A, Adhanom IB, Folmer E. Embodied Third-Person Virtual Locomotion using a Single Depth Camera. In: Proceedings of Graphics Interface 2021. pp. 210–219. Canadian Information Processing Society (2021). DOI 10.20380/GI2021.24.
- [32] KPMG: Neue Dimensionen der Realität. Eine Analyse der Potenziale von Virtual und Augmented Reality in Unternehmen. (2016).
- [33] HPI: Technologiemonitoring Steckbrief VR / AR. (2019).
- [34] Kim E, Shin G. User discomfort while using a virtual reality headset as a personal viewing system for text-intensive office tasks. Ergonomics. 64, 891–899 (2021). DOI 10.1080/00140139.2020.1869320.
- [35] Harmaz K. Constructing grounded theory: A practical guide through qualitative analysis. Sage (2006).
- [36] VDI Verein Deutscher Ingenieure: VDI Richtlinie 3633, Blatt 1, (2014).
- [37] Rossit DA, Vásquez ÓC, Tohmé F, Frutos M, Safe MD. The Dominance Flow Shop Scheduling Problem. Electron. Notes Discret. Math. 69, 21–28 (2018). DOI 10.1016/j.endm.2018.07.004.
- [38] Baker K, Trietsch D. Flow Shop Scheduling. In: Principles of Sequencing and Scheduling. pp. 283–317. Wiley (2018).
- [39] Tellache NEH, Boudhar M. Flow shop scheduling problem with conflict graphs. Ann. Oper. Res. 261, 339–363 (2018). DOI 10.1007/s10479-017-2560-x.
- [40] Lawler EL, Lenstra JK., Rinnooy Kan AHG, Shmoys DB. Chapter 9 Sequencing and scheduling: Algorithms and complexity. Presented at the (1993). DOI 10.1016/S0927-0507(05)80189-6.
- [41] Kassu J, Eshetie B. Job Shop Scheduling Problem for Machine Shop with Shifting Heuristic Bottleneck. Glob. J. Res. Eng. 158, (2015).
- [42] Li Y, Goga K, Tadei R, Terzo O. Production Scheduling in Industry 4.0. DOI 10.1007/978-3-030-50454-0_34.
- [43] Koruca H, Aydemir E. A Priority Rule Based Production Scheduling Module on Faborg-Sim Simulation Tool. Gazi Univ. J. Sci. 27, 1143–1155 (2014).
- [44] Swamidass PM (ed). Priority Scheduling Rules. In: Encyclopedia of Production and Manufacturing Management. pp. 527–528. Springer US, Boston, MA (2000). DOI 10.1007/1-4020-0612-8_708.
- [45] Dexter LA. Elite and Specialized Interviewing. Northwestern University Press (1970).
- [46] Glaser BG, Strauss AL. The Discovery of Grounded Theory: Strategies for Qualitative Research. Aldine Transaction (1967).



EUROSIM Data and Quick Info

		ASIM SST 2024 27. Symposium Simulationstechnik 27 th Symposium Simulation Technique 4. - 6. September 2024, München, Univ. BW, Deutschland www.asim-gi.org	
	SIMS 2024 – 65. Int. Conference September 2024, Scandinavia www.scansims.org	I3M 2024 International Multidisciplinary Modeling & Simulation Multiconference www.msc-les.org/i3m2024	
	 EUROSIM CONGRESS 2026 July 2026, Italy www.eurosim.info	 Winter Simulation Conference 2024, December 15-18, 2024 Orlando, FL, USA www.wintersim.org	
	ASIM FG Workshop SUG 2024 Simulation in Umwelt-und Geowissenschaften 10.4.-12.4., Leipzig, Deutschland	ASIM Workshops GMMS/STS/EDU 2024 in ASIM SST 2024, September 2024, München www.asim-gi.org	



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 modelling and simulation in industry, research, and development – by publication and conferences.

www.eurosim.info

EUROSIM members may be national simulation societies and regional or international societies and groups dealing with modelling and simulation.

Full Members are ASIM, CEA-SMSG, CSSS, DBSS, KASIM, LIOPHANT, LSS, PTSK, NSSM, SIMS, SLOSIM, UKSIM. Observer Members are ALBSIM and ROMSIM. Former Members (societies in re-organisation) are: CROSSIM, FRANCOSIM, HSS, ISCS.

EUROSIM is governed by a **Board** consisting of one representative of each member society, president, past president, and SNE representative.

President	Agostino Bruzzone (LIOPHANT) agostino@itim.unige.it
Past President	M. Mujica Mota (DBSS), m.mujica.mota@hva.nl
Secretary	Marina Massei (LIOPHANT), massei@itim.unige.it
Treasurer	Felix Breitenecker (ASIM) felix.breitenecker@tuwien.ac.at
Webmaster	Irmgard Husinsky (ASIM), irmgard.husinsky@tuwien.ac.at

SNE SNE – Simulation Notes Europe is EUROSIM’s membership journal with peer reviewed scientific contributions about all areas of modelling and simulation, including new trends as big data, cyber-physical systems, etc.

The EUROSIM societies distribute e-SNE in full version to their members as official membership journal. The basic version of e-SNE is available with open access. Publishers are EUROSIM, ARGESIM and ASIM,

www.sne-journal.org

office@sne-journal.org

SNE-Editor: Felix Breitenecker (ASIM)

felix.breitenecker@eic@sne-journal.org

EUROSIM Congress and Conferences

Each year a major EUROSIM event takes place, as the EUROSIM CONGRESS organised by a member society, SIMS EUROSIM Conference, and MATHMOD Vienna Conference (ASIM).

On occasion of the EUROSIM Congress 2023, the 11th *EUROSIM Congress* in Amsterdam, July, 2023, a new EUROSIM president has been elected: we welcome Agostino Bruzzone, well known simulationist, as new president. His society LIOPHANT will organize the next EUROSIM Congress in 2026 in Italy.

Furthermore, EUROSIM Societies organize local conferences, and EUROSIM co-operates with the organizers of I3M Conference and WinterSim Conference Series.



EUROSIM Member Societies



ASIM German Simulation Society Arbeitsgemeinschaft Simulation

ASIM is the association for simulation in the German speaking area, servicing mainly Germany, Switzerland and Austria.

President	Felix Breiteneker, <i>felix.breiteneker@tuwien.ac.at</i>
Vice President	Sigrid Wenzel, <i>s.wenzel@uni-kassel.de</i> Thorsten Pawletta, <i>thorsten.pawletta@hs-wismar.de</i> Andreas Körner, <i>andreas.koerner@tuwien.ac.at</i>

ASIM is organising / co-organising the following international conferences: ASIM SPL Int. Conference 'Simulation in Production and Logistics' (biannual), ASIM SST 'Symposium Simulation Technique' (biannual), MATHMOD Int. Vienna Conference on Mathematical Modelling (triennial). Furthermore, ASIM is co-sponsor of WSC - Winter Simulation Conference and of the *I3M* and conference series.

ASIM Working Committees

GMMS: Methods in Modelling and Simulation

U. Durak, *umut.durak@dlr.de*

SUG: Simulation in Environmental Systems

J. Wittmann, *wittmann@informatik.uni-hamburg.de*

STS: Simulation of Technical Systems

W. Commerell, *commerell@hs-uhl.de*

SPL: Simulation in Production and Logistics

S. Wenzel, *s.wenzel@uni-kassel.de*

EDU: Simulation and Education

A. Körner, *andreas.koerner@tuwien.ac.at*

Working Group Big Data: Data-driven Simulation in

Life Sciences, N. Popper, *niki.popper@dwh.at*

Other Working Groups: Simulation in Business Administration, in Traffic Systems, for Standardisation, etc.

Contact Information

www.asim-gi.org

info@asim-gi.org, admin@asim-gi.org

ASIM – Inst. of Analysis and Scientific Computing,
TU Wien, Wiedner Hauptstraße 8-10, 1040 Vienna,
Austria

CEA-SMSG – Spanish Modelling and Simulation Group

CEA is the Spanish Society on Automation and Control. The association is divided into national thematic groups, one of which is centered on Modeling, Simulation and Optimization (CEA-SMSG).

President	José L. Pitarch, <i>jlpitarch@isa.upv.es</i>
Vice President	Juan Ignacio Latorre, <i>juanignacio.latorre@unavarra.es</i>

Contact Information

www.ceautomatica.es/grupos/

simulacion@cea-ifac.es

CEA-SMSG / Emilio Jiménez, Department of Electrical Engineering, University of La Rioja, San José de Calasanz 31, 26004 Logroño (La Rioja), Spain



CSSS – Czech and Slovak Simulation Society

CSSS is the Simulation Society with members from the two countries: Czech Republic and Slovakia. The CSSS history goes back to 1964.

President	Michal Štepanovský <i>michal.stepanovsky@fit.cvut.cz</i>
Vice President	Mikuláš Alexík, <i>alexik@frtk.fri.utc.sk</i>

Contact Information

cssim.cz

michal.stepanovsky@fit.cvut.cz

CSSS – Český a Slovenský spolek pro simulaci systémů, Novotného lávka 200/5,
11000 Praha 1, Česká republika



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.

President	M. Mujica Mota, <i>m.mujica.mota@hva.nl</i>
Vice President	A. Heemink, <i>a.w.heemink@its.tudelft.nl</i>
Secretary	P. M. Scala, <i>paolo.scala@fedex.com</i>



Contact Information

www.DutchBSS.org

a.w.heemink@its.tudelft.nl

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

KA-SIM Kosovo Simulation Society

The Kosova Association for Modeling and Simulation (KA-SIM) is closely connected to the University for Business and Technology (UBT) in Kosovo.

President	Edmond Hajrizi, ehajrizi@ubt-uni.net
Vice President	Muzafer Shala, info@ka-sim.com

Contact Information

www.ubt-uni.net

ehajrizi@ubt-uni.net

Dr. Edmond Hajrizi
Univ. for Business and Technology (UBT)
Lagjja Kalabria p.n., 10000 Prishtina, Kosovo



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, courses and internships focused on M&S applications.

President	A.G. Bruzzone, agostino@itim.unige.it
Director	E. Bocca, enrico.bocca@liophant.org

Contact Information

www.liophant.org

info@liophant.org

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

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.

President	Artis Teilans, Artis.Teilans@rtu.lv
Vice President	Oksana Kuznecova, Oksana.Kuznecova@rtu.lv

Contact Information

www.itl.rtu.lv/imb/

Artis.Teilans@rtu.lv, Egils.Ginters@rtu.lv

LSS, Dept. of Modelling and Simulation, Riga Technical University, Kalku street 1, Riga, LV-1658, Latvia



NSSM – National Society for Simulation Modelling (Russia)

NSSM – The National Society for Simulation Modelling (Национальное Общество Имитационного Моделирования – НОИМ) was officially registered in Russia in 2011.

President	R. M. Yusupov, yusupov@iias.spb.su
Chairman	A. Plotnikov, plotnikov@sstc.spb.ru

Contact Information

www.simulation.su

yusupov@iias.spb.su

NSSM / R. M. Yusupov, St. Petersburg Institute of Informatics and Automation RAS, 199178, St. Petersburg, 14th line, h. 39

PTSK – Polish Society for Computer Simulation

PTSK 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.

President	Tadeusz Nowicki, Tadeusz.Nowicki@wat.edu.pl
Vice President	Leon Bobrowski, leon@ibib.waw.pl

Contact Information

www.ptsk.pl

leon@ibib.waw.pl

PSCS, 00-908 Warszawa 49, ul. Gen. Witolda Urbanowicza 2, pok. 222



SIMS – Scandinavian Simulation Society

SIMS is the Scandinavian Simulation Society with members from the five Nordic countries Denmark, Finland, Norway, Sweden and Iceland. The SIMS history goes back to 1959.

President	Tiina Komulainen, <i>tiina.komulainen@oslomet.no</i>
Vice President	Erik Dahlquist, <i>erik.dahlquist@mdh.se</i>

Contact Information

www.scansims.org

vadime@wolfram.com

Vadim Engelson, Wolfram MathCore AB,
Teknikringen 1E, 58330, Linköping, Sweden



SLOSIM – Slovenian Society for Simulation and Modelling

The Slovenian Society for Simulation and Modelling was established in 1994. It promotes modelling and simulation approaches to problem solving in industrial and in academic environments by establishing communication and cooperation among corresponding teams.

President	Goran Andonovski, <i>goran.andonovski@fe.uni-lj.si</i>
Vice President	Božidar Šarler, <i>bozidar.sarler@fs.uni-lj.si</i>

Contact Information

www.slosim.si

slosim@fe.uni-lj.si, vito.logar@fe.uni-lj.si

SLOSIM, Fakulteta za elektrotehniko, Tržaška 25,
SI-1000, Ljubljana, Slovenija

UKSIM - United Kingdom Simulation Society

The UK Modelling & Simulation Society (UKSim) is the national UK society for all aspects of modelling and simulation, including continuous, discrete event, software and hardware.

President	David Al-Dabass, <i>david.al-dabass@ntu.ac.uk</i>
Secretary	T. Bashford, <i>tim.bashford@uwtsd.ac.uk</i>

Contact Information

uksim.info

david.al-dabass@ntu.ac.uk

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

Observer Members

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.

Contact Information

florin_h2004@yahoo.com

ROMSIM / Florin Hartescu, National Institute for Research in Informatics, Averescu Av. 8 – 10, 011455 Bucharest, Romania

ALBSIM – Albanian Simulation Society

The Albanian Simulation Society has been initiated at the Department of Statistics and Applied Informatics, Faculty of Economy at the University of Tirana, by Prof. Dr. Kozeta Sevrani.

Contact Information

kozeta.sevrani@unitir.edu.al

Albanian Simulation Goup, attn. Kozeta Sevrani, University of Tirana, Faculty of Economy, rr. Elbasanit, Tirana 355, Albania

Former Societies / Societies in Re-organisation

- CROSSIM – Croatian Society for Simulation Modelling
Contact: Tarzan Legović, *Tarzan.Legovic@irb.hr*
- FrancoSim – Société Francophone de Simulation
- HSS – Hungarian Simulation Society
Contact: A. Gábor, *andrasi.gabor@uni-bge.hu*
- ISCS – Italian Society for Computer Simulation

The following societies have been formally terminated:

- MIMOS – Italian Modeling & Simulation Association; terminated end of 2020.

ARGESIM is a non-profit association generally aiming for dissemination of information on system simulation – from research via development to applications of system simulation. **ARGESIM** is closely co-operating with **EUROSIM**, the Federation of European Simulation Societies, and with **ASIM**, the German Simulation Society.

ARGESIM is an 'outsourced' activity from the *Mathematical Modelling and Simulation Group* of TU Wien, there is also close co-operation with **TU Wien** (organisationally and personally).

→ www.argesim.org

→ office@argesim.org

→ ARGESIM/Math. Modelling & Simulation Group,
Inst. of Analysis and Scientific Computing, TU Wien
Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria
Attn. Prof. Dr. Felix Breitenecker

ARGESIM is following its aims and scope by the following activities and projects:

- Publication of the scientific journal **SNE – Simulation Notes Europe** (membership journal of **EUROSIM**, the *Federation of European Simulation Societies*) – www.sne-journal.org
- Organisation and Publication of the **ARGESIM Benchmarks** for *Modelling Approaches and Simulation Implementations*
- Publication of the series **ARGESIM Reports** for monographs in system simulation, and proceedings of simulation conferences and workshops
- Publication of the special series **FBS Simulation – Advances in Simulation / Fortschrittsberichte Simulation** - monographs in co-operation with **ASIM**, the German Simulation Society
- Support of the Conference Series **MATHMOD Vienna** (triennial, in co-operation with **EUROSIM**, **ASIM**, and **TU Wien**) – www.mathmod.at
- Administration of **ASIM** (German Simulation Society) and administrative support for **EUROSIM** www.eurosim.info
- Simulation activities for TU Wien

ARGESIM is a registered non-profit association and a registered publisher: **ARGESIM Publisher Vienna**, root ISBN 978-3-901608-xx-y and 978-3-903347-xx-y, root DOI 10.11128/z...zz.zz. Publication is open for **ASIM** and for **EUROSIM Member Societies**.

The scientific journal **SNE – Simulation Notes Europe** provides an international, high-quality forum for presentation of new ideas and approaches in simulation – from modelling to experiment analysis, from implementation to verification, from validation to identification, from numerics to visualisation – in context of the simulation process. **SNE** puts special emphasis on the overall view in simulation, and on comparative investigations.

Furthermore, **SNE** welcomes contributions on education in/for/with simulation.

SNE is also the forum for the **ARGESIM Benchmarks** on *Modelling Approaches and Simulation Implementations* publishing benchmarks definitions, solutions, reports and studies – including model sources via web.

SNE Editorial Office /ARGESIM

→ www.sne-journal.org

office@sne-journal.org, eic@sne-journal.org

Johannes Tanzler (*Layout, Organisation*)
Irmgard Husinsky (*Web, Electronic Publishing*)
Felix Breitenecker *EiC (Organisation, Authors)*

ARGESIM/Math. Modelling & Simulation Group,
Inst. of Analysis and Scientific Computing, TU Wien
Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria

SNE, primarily an electronic journal, follows an open access strategy, with free download in a basic version (B/W, low resolution graphics). **SNE** is the official membership journal of **EUROSIM**, the *Federation of European Simulation Societies*. Members of (most) **EUROSIM Societies** are entitled to download the full version of e-**SNE** (colour, high-resolution graphics), and to access additional sources of benchmark publications, model sources, etc. (group login for the 'publication-active' societies; please contact your society). Furthermore, **SNE** offers **EUROSIM Societies** a publication forum for post-conference publication of the society's international conferences, and the possibility to compile thematic or event-based **SNE Special Issues**.

Simulationists are invited to submit contributions of any type – *Technical Note, Short Note, Project Note, Educational Note, Benchmark Note*, etc. via **SNE's** website:

→ www.sne-journal.org,



Preliminary Announcement
Save the date

ASIM 2024

27. Symposium Simulationstechnik
4. - 6. September 2024



The scope of the *ASIM Symposium Simulationstechnik* – also including the *workshop of the working groups GMMS, STS, and EDU* – covers basics, methods, and tools of modeling and simulation as well as all areas of application (from engineering sciences to computer science, production and logistics, bio-, environmental and geosciences, climate and ecosystem, up to training and education in modeling and simulation).

The **Scientific Programme** of **ASIM 2024** includes *Invited Lectures*, *Contributed Lectures* in parallel sessions, and *Tutorials*. Of course a **Social Programme** will be organized.

Conference languages are German and English.

More information will be available soon at: www.asim-gi.org

www.sne-journal.org
www.argesim.org

