Jonas Gesenhues*, Marc Hein, Maike Ketelhut, Moriz Habigt, Daniel Rüschen, Mare Mechelinck, Thivaharan Albin, Steffen Leonhardt, Thomas Schmitz-Rode, Rolf Rossaint, Rüdiger Autschbach and Dirk Abel

Benefits of object-oriented models and ModeliChart: modern tools and methods for the interdisciplinary research on smart biomedical technology

DOI 10.1515/bmt-2016-0074
Received March 23, 2016; accepted November 28, 2016; online first January 25, 2017

Abstract: Computational models of biophysical systems generally constitute an essential component in the realization of smart biomedical technological applications. Typically, the development process of such models is characterized by a great extent of collaboration between different interdisciplinary parties. Furthermore, due to the fact that many underlying mechanisms and the necessary degree of abstraction of biophysical system models are unknown beforehand, the steps of the development process of the application are iteratively repeated when the model is refined. This paper presents some methods and tools to facilitate the development process. First, the principle of object-oriented (OO) modeling is presented and the advantages over classical signal-oriented modeling are emphasized. Second, our self-developed simulation tool ModeliChart is presented. ModeliChart was designed specifically for clinical users and allows independently performing *in silico* studies in real time including intuitive interaction with the model. Furthermore, ModeliChart is capable of interacting with hardware such as sensors and actuators. Finally, it is presented how optimal control methods in combination with OO models can be used to realize clinically motivated control applications. All methods presented are illustrated on an exemplary clinically oriented use case of the artificial perfusion of the systemic circulation.

Keywords: biophysical systems; hardware interaction; interdisciplinary collaboration; Modelica; ModeliChart; object-oriented modeling; optimal control; Optimica; simulation.

***Corresponding author: Jonas Gesenhues,** Institute of Automatic Control, RWTH Aachen, Steinbachstraße 54, 52056 Aachen, Germany, E-mail: j.gesenhues@irt.rwth-aachen.de. http://orcid.org/0000-0003-4476-8727
Marc Hein, Moriz Habigt, Mare Mechelinck and Rolf Rossaint: Department of Anesthesiology, RWTH Aachen University Hospital, Aachen, Germany
Maike Ketelhut, Thivaharan Albin and Dirk Abel: Institute of Automatic Control, RWTH Aachen University, Aachen, Germany
Daniel Rüschen and Steffen Leonhardt: Philips Chair for Medical Information Technology, Helmholtz-Institute for Biomedical Engineering, RWTH Aachen University, Aachen, Germany
Thomas Schmitz-Rode: Institute of Applied Medical Engineering, Helmholtz-Institute for Biomedical Engineering, RWTH Aachen University, Aachen, Germany
Rüdiger Autschbach: Department of Thoracic and Cardiovascular Surgery, University Hospital RWTH Aachen, Aachen, Germany

Introduction

The word smart in “Smart Life Support” – the overall headline of this special issue – refers to the application of methods and strategies that explicitly exploit knowledge of the system of interest, specifically the physiology of living (human) systems. The motivation for applying smart methods and strategies is manifold. Typical examples include finding improved treatment strategies (e.g. the prediction of the outcome of specific interventions) or creating new control and automation strategies for technological assist devices. To exemplify the latter, a smart controller for a blood pump aims at achieving specific states within the physiological system – as opposed to classical controllers that typically only act on the difference between the desired and the actual value of a chosen technical variable (e.g. a speed controller).

In either motivation, two key aspects arise in the development of smart approaches. The first key aspect is the distinct need for an intense collaboration between the interdisciplinary parties involved (interdisciplinary aspect). Those parties include specialists for the considered aspect of the physiological system as well as technological, simulation and control specialists. Within the

scope of this paper, the involved parties are represented through a physician and an engineer. The second key aspect is the development and use of adequate computational models of the physiological system of interest (e.g. models of the cardiovascular, the pulmonary system or the metabolic system, model-based aspect). Major challenges in the development of such physiological system models are that not all underlying mechanisms are fully understood yet and the adequate degree of abstraction is not known *a priori*. Consequently, the development of models is an iterative process and tightly bound to the desired application.

The aim of this paper is to present innovative methods, tools and techniques to facilitate both key aspects. A special emphasis of this paper will be on the introduction of our free self-developed simulation tool ModeliChart which has been specifically designed for the application in biomedical research and engineering.

The presented methods are quite general and thus applicable to various physiological and technical system fields. Therefore, this paper addresses groups researching all sorts of biomedical topics. Nevertheless, to demonstrate the methods and tools, throughout all sections of this paper, a simplified exemplary application exhibiting common problems and questions, which may similarly arise in other fields, is constructed; considered is the field of artificial perfusion of the systemic circulation. The systemic circulation includes the large arteries (such as the aorta) which have the ability to distend with rising blood pressure and to recoil with falling blood pressure (physiological compliance). Within the living system, the physiological function of the arterial vessels is to act as a damper on the pulsatile pressure over the cardiac cycle. However, within the scope of artificial perfusion, the heart is replaced by a technical blood pump device (referred to simply as pump in the following). In this regard, two questions arise:

1. What are the emerging flow dynamics (i.e. the trajectory of the pressure and flow signals) as a result of the pump's action?
2. How can the pump be operated to achieve specified flow dynamics?

The paper is organized along the mentioned key aspects and questions: first, it addresses the model-based aspect. Hence, a general introduction of the principles of object-oriented (OO) modeling is provided (section Object-oriented modeling). Afterward, ModeliChart is introduced and its capabilities to facilitate the interdisciplinary aspect as well as to answer the first question are illustrated (section Simulation with ModeliChart). ModeliChart can also be used to interface hardware components. As an example, our setup to drive a mock circulatory loop (MCL) for experiments involving blood pumps is shown (section ModeliChart and hardware interaction). Considering the second question, the use of OO models within control applications is demonstrated; using an optimal control approach, the necessary input for an ideal flow pump to achieve a specified pulsatile profile is calculated (section Object-oriented model-based optimal control). Finally, the ideal flow pump is exchanged for a real existing blood pump. Therefore, an *in-vitro* experiment realized through ModeliChart and the MCL is conducted to verify whether the results of the optimal control problem (section Mock circulatory loop tests of the optimal control solution) can be achieved using a real pump. Results of all sections are presented. The paper concludes with a discussion on possible benefits and limitations of the presented methods and with proposals for further enhancements.

In all of the following sections, a simple lumped-parameter (spatially zero-dimensional) model of the systemic circulation is considered. The large arteries and arterioles are modeled using a three-element windkessel (TEW) model, originally described by Westerhof [19]. The TEW consists of two hydraulic resistances and a compliance element (Figure 1). Over a hydraulic resistance, the blood pressure drop ΔP is proportional to the (blood) flow Q ($\Delta P = R \cdot Q$, with R being the proportionality factor of the hydraulic resistance). A compliance component is characterized with the time derivate of the pressure $\dot{P}$ being proportional to the flow into the component ($\dot{P} \cdot C = Q$, with C being the proportionality factor of the compliance). Providing that the (blood) volume V in the compliance component is zero at zero pressure (i.e. the pressure within the compliance component is equal to the surrounding pressure), it can also be written as the pressure being proportional to the blood in the component ($P \cdot C = V$). It is assumed that the venous system can be approximated by applying a constant boundary pressure to the outlet of the TEW ($P_{W,out} = \text{const.} = P_B$). For convenience, all absolute pressure values will be referenced to P_B, i.e. $P_B = 0$ mm Hg.

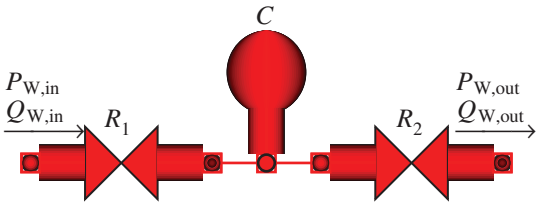

Figure 1: The three-element windkessel (TEW) model.

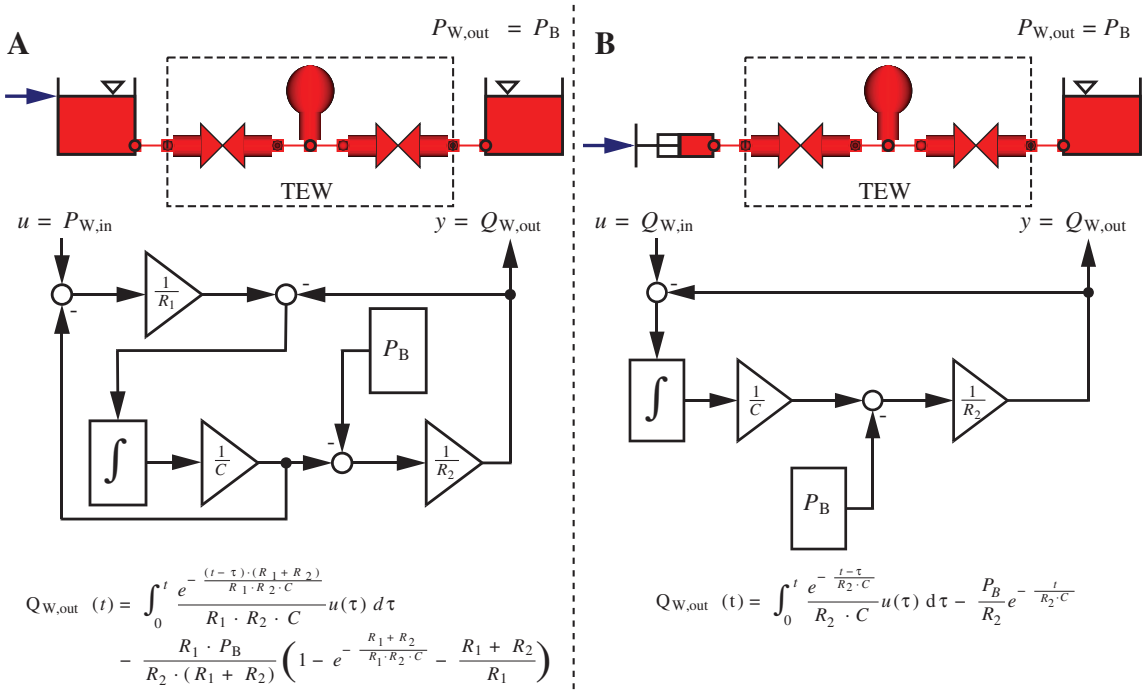

Figure 2: Comparison between the circulation system driven by ideal pressure (A) and ideal flow (B) source. Depicted are a graphical object-oriented component-based representation according to Listing 1 (top), a signal-oriented block diagram (middle) and a mathematical expression (bottom) for the outlet flow.

Materials and methods

OO modeling

Within the field of developing computational models of (bio-) physical systems, OO modeling is a paradigm that comprises a set of principles which is not found in classical signal-based modeling approaches. Specifically, those principles include component-based modeling and acausal modeling, which will be briefly introduced below.

Component-based modeling is the idea of composing complex large scale models out of small and independent components. Using signal-based modeling approaches, the first necessary step to perform is to determine the cause and effect chain, i.e. deciding on the inputs and outputs of a component. However, this can lead to different representations of the same component. For example, a hydraulic resistance (R) can be modeled to be driven by a pressure difference ($u_{Pressure} = \Delta P$). Here, the resulting flow is a dependent variable ($y_{Flow} = u_{Pressure}/R$). However, the resistance could also be driven by flow ($u_{Flow} = Q$). In this case, the resulting pressure is a dependent variable ($y_{Pressure} = u_{Flow} \cdot R$). Consequently, two different model representations of the same component are possible. Based on the cause on hand, it needs to be chosen which representation is the right one. However, in larger models, this can become more and more difficult. Besides, the algebraic transformation of the differential equation governing the component can be complex. Even for simple models such as the model of the systemic circulation based on a TEW and a constant outlet pressure, those difficulties related to signal-oriented modeling become apparent; Figure 2 shows a comparison of signal-oriented models when the systemic circulation model is driven by either an ideal pressure source (A) or by an ideal flow source (B). In both cases, the outlet flow has been chosen as the output variable of interest. As can be seen from the corresponding block diagrams and the mathematical expressions in Figure 2, the individual representations of the overall system differ significantly from each other.

The idea of acausal modeling is to take the tasks of deciding on the cause and effect and rearranging equations off the developer. Acausal OO models are formulated using sophisticated modeling languages (similar to programming languages) and development tools. We use Modelica® which is a free, non-proprietary OO modeling language maintained by the Modelica Association [10]. In Modelica, the governing equation of components can be written without specifying inputs and outputs and consequently without the need for rearranging equations. Based on how individual components are connected and the boundary conditions of the overall system, the equations are automatically rearranged in a compilation process before the model is simulated. Today, commercial and free Modelica compilers exist, for example Dymola® (Dassault Systems, Vélizy-Villacoublay, France), Wolfram SystemModeler® (Wolfram Research, Champaign, IL, USA) and the open-source initiatives OpenModelica.org and JModelica.org.

Besides component-based and acausal modeling principles, the Modelica language also comprises concepts known from OO programming languages such as inheritance, allowing for a high degree of reusability of existing components. This has led to the development of component libraries for various physical domains. The Modelica Association maintains a Modelica Standard library, which contains components of various physical domains (mechanical, electrical, hydraulic, thermal and many more). To cater for components to build models of the cardiovascular systems, our open-source Modelica library HumanLib contains components of the mammalian cardiovascular system [9]. Listing 1 demonstrates how the systemic circulation can be modeled in Modelica using components of the HumanLib. The listing contains three different models; the first model is an implementation of the TEW. The extend keyword is used to inherit from another

```
model ThreeElemWindkessel
 import HumanLib.Basics.*;
 import HumanLib.Vessels.*;
 extends Interfaces.BloodStreamTwoPin;
 Compliance C(V(start=0, fixed=true), C=1);
 Resistance R1(R=0.1);
 Resistance R2(R=1);
equation
 connect(C.cnBloodStream, R1.cnStreamOut);
 connect(R1.cnStreamIn, cnStreamIn);
 connect(C.cnBloodStream, R2.cnStreamIn);
 connect(cnStreamOut, R2.cnStreamOut);
end ThreeElemWindkessel;

model PressureDrivenSystem
 import HumanLib.Basics.*;
 replaceable ThreeElemWindkessel ArterialSystem
    constrainedby Interfaces.BloodStreamTwoPin;
 Sources.ConstantPressureBoundary P_B(p=0);
 Sources.PressureSource_Variable PressureSource;
 input Real u;
equation
 connect(PressureSource.cnBloodStream,
         ArterialSystem.cnStreamIn);
 connect(ArterialSystem.cnStreamOut,
         P_B.cnBloodStream);
 PressureSource.P=u;
end PressureDrivenSystem;

model FlowDrivenSystem
 import HumanLib.Basics.*;
 replaceable ThreeElemWindkessel ArterialSystem
    constrainedby Interfaces.BloodStreamTwoPin;
 Sources.ConstantPressureBoundary P_B(p=0);
 Sources.FlowSource_Variable FlowSource;
 input Real u;
equation
 connect(Flow_Source.cnBloodStream,
         ArterialSystem.cnStreamIn);
 connect(ArterialSystem.cnStreamOut,
         P_B.cnBloodStream);
 FlowSource.Q=u;
end FlowDrivenSystem;
```

Listing 1: Modelica code of the pressure-driven and flow-driven systems depicted in Figure 2.

component. Here, inheriting from the BloodStreamTwoPin provides the in- and outlet connectors for the TEW model. Next, the components for the TEW are defined. In the equation section, the components are connected with each other. The remaining two models are the OO pressure and flow-driven systemic circulation according to Figure 2. The replaceable keyword indicates that the TEW component can be exchanged for a different component. Replacing components within models is further referred to within the Discussion section. A stripped-down, but yet complete version to realize the models in Listing 1 of the HumanLib can be found as online supplement. Other examples of using Modelica for modeling human physiology in general and the cardiovascular system in particular include the Physiolibrary [15] and works by Heinke et al. [13].

Simulation with ModeliChart

ModeliChart is a simulation software tool for Microsoft Windows® (Redmond, WA, USA) that has been developed at our institute. It is freely available upon request. ModeliChart enables the physician to independently perform *in silico* studies and to boost the efficiency of collaborative efforts toward creating new and improve existing models; despite the potential benefits of OO models, the simulation expertise required and the need for advanced and possibly expensive software constitute a significant constriction of the workflow. The original motivation for the development of ModeliChart has been to enable the application expert (in our case the physician) to investigate models without requiring simulation expertise. ModeliChart provides a simple and intuitive stand-alone user interface that provides all the necessary features to perform simulation (*in silico*) experiments. At the same time, ModeliChart automatically takes care of all the involved technical details in the background. With ModeliChart, the collaborative workflow starts with the engineer creating a draft of a simulation model and sending it to the physician. The physician then opens the model in ModeliChart and independently performs simulations to assess the underlying models, controllers, et cetera. The focus of ModeliChart is on the simulation in real time (providing that the computational effort necessary to simulate the loaded models can be handled) as we found that real-time simulations are most intuitive for clinical experts. During the simulation, the physician has the opportunity to manipulate independent model variables such as free parameters and inputs. The results of such variable manipulation interventions can be immediately observed. The physician has the opportunity to save the simulation results and chosen variable values.

To be independent of specific vendors or modeling tools, ModeliChart is based on the second version of the functional mockup interface (FMI) co-simulation standard. The FMI standard is an open standard that has been initiated to facilitate the exchange of simulation models [8]. It is maintained and developed by companies and research institutes. The first version has been published in 2010; the second and at this time most recent version of the standard on which ModeliChart is based on has been released in mid-2014. A simulation model complying with the FMI standard is called a functional mockup unit (FMU). A FMU is distributed as a single file. The file contains the definition of the model as well as compiled C-code of the model including the necessary numerical solver. A FMU can be directly opened in ModeliChart and simulated as described before. More and more modeling tools support the generation of FMUs, including all the Modelica compilers mentioned above. However, also non-Modelica-based modeling tools such as Simulink® [The Mathworks, Inc., (Natick, MA, USA) using third party add-ons] support FMU export. Therefore, ModeliChart is not limited to Modelica models. A tabular summary of the tools supporting the FMI standard can be found at fmi-standard.org.

ModeliChart is not limited to the simulation of a single FMU but supports the parallel simulation of multiple FMUs. Individual channels of the FMUs can be linked to each other, thus enabling the set of FMUs to interact not only with the user but also with each other. As a typical example, a possible setup could consist of two FMUs of a system to be controlled (typically termed “plant” in control theory) which could have been developed in Modelica and a controller developed in Simulink®. In this regard, ModeliChart serves as a convenient environment for the co-simulation of multiple models developed with different tools allowing for flexible system simulation, also known as model-in-the-loop (MiL) studies. The setup of all the FMUs including the channel links can be saved in a single file and shared with other users.

To provide an example for typical usage of ModeliChart, Figure 3 shows a screenshot of a simulation that the physician might have set up to gain insight into the effects of a pulsatile perfusion of the arterial system described in the previous section. The setup consists of two FMUs. The first FMU contains the model of the pressure source driven arterial system (Figure 2A). The second FMU is a signal

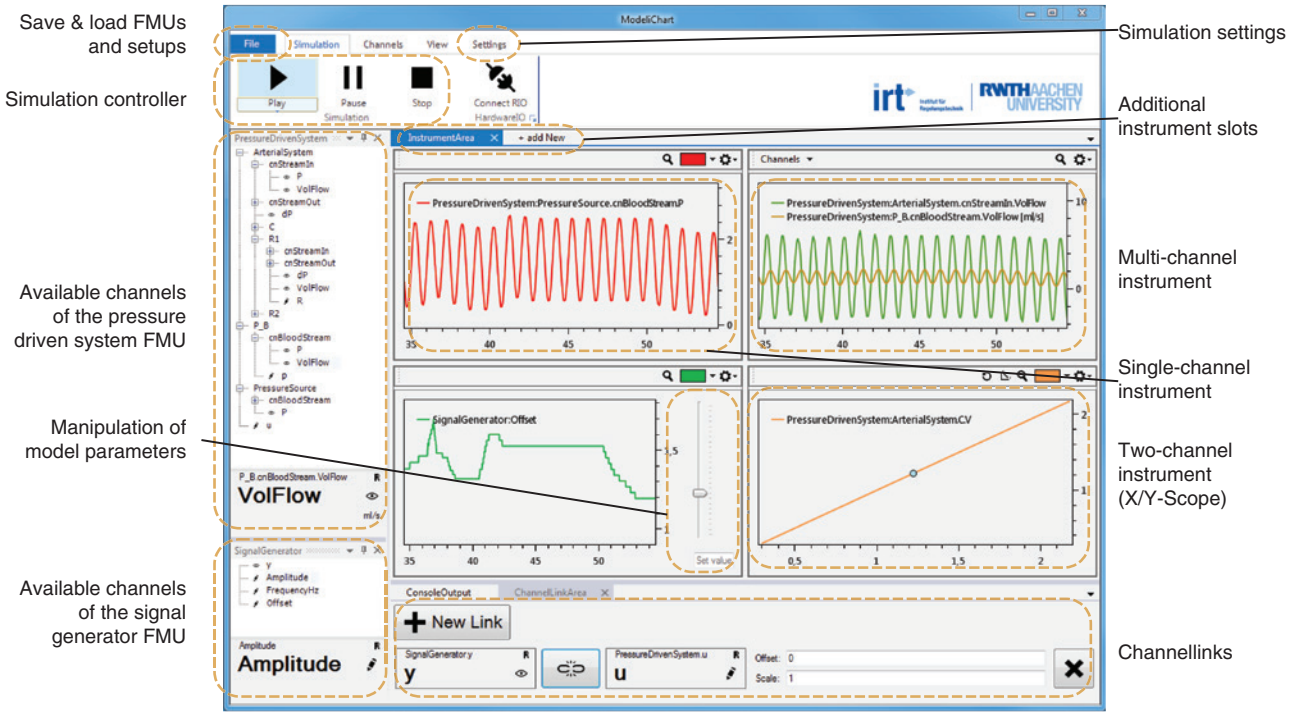

Figure 3: Screenshot of a simulation in ModeliChart.

generator that generates a sinusoidal signal with freely adjustable frequency, amplitude and offset. The output of the signal generator FMU is connected to the input of the ideal pressure source. On the left hand side, all the available channels are displayed. The user can drag and drop any channel onto an instrument within the instrument area. Different instruments are available: besides displaying one or multiple channels over time, a two-axis instrument allows choosing two channels to be plotted on the x and y axes. For example, within the field of cardiac research, this allows for real-time plots of (e.g. ventricular) pressure-volume diagrams. In Figure 3, the pressure and the volume channels of the compliance are plotted as a two-axes instrument in the lower right corner. Here, as the proportional parameter C of the compliance element is constant, this plot is a straight line with a blue marker indicating the current channel values.

Channels representing independent variables (inputs or parameters) of the model can be set to new values by the user either by using a slider control or by typing a specific value. In Figure 3, the set value for the offset of the signal generator (plotted in the lower left corner) is changed to new values. The instrument in the upper left corner shows the reaction of the actual value of the pressure source. The resulting flows into the arterial system and into the pressure boundary are plotted within the instrument in the upper right corner. The section at the bottom shows the graphical representation of the channel links in ModeliChart. The channel links also offer the possibility of a linear transformation of the linked values. This can be useful to perform unit conversions.

ModeliChart and hardware interaction

Although initially designed to be an independent but flexible simulation environment for MiL studies, several use cases have been identified that motivated the extension of ModeliChart to be able to interact with real hardware such as (medical) sensors and actuators. Through hardware interaction, the capabilities of ModeliChart are increased to enable software-in-the-loop (SiL) and hardware-in-the-loop (HiL) experiments. For example, if a controller for an artificial blood pump is to be designed, SiL refers to connecting the real pump hardware to ModeliChart. In this case, ModeliChart simulates the control algorithm in real time and allows for the online tuning of controller parameters. In contrast, HiL refers to connecting the controller implemented on the intended target hardware to ModeliChart to test the controller without a real pump. Here, ModeliChart simulates the behavior of the pump based on the input of the controller. All in all, ModeliChart in combination with hardware interaction can be used during all steps of the V-model within field of rapid control prototyping (for more information on RCP see [1]).

The realization of hardware interaction integration consists of two components (Figure 4). The first component is a hardware unit that provides the necessary interface to connect to the target hardware. An embedded solution based on the myRIO FPGA/microprocessor platform (National Instruments™, Austin, TX, USA) housed in a 3D printed case has been developed (Figure 5). As the vast majority

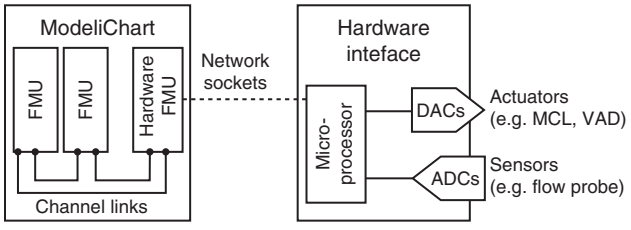

Figure 4: Concept of the hardware interaction with ModeliChart.

Figure 5: Hardware interface unit to be used with ModeliChart.

Figure 6: The mock circulatory loop used in this study as an exemplary hardware device to be driven by ModeliChart.

of the medical and (bio-) sensor target hardware we are working with uses analog voltage signals for communication, the hardware unit includes analog-digital converters for inputs to ModeliChart and digital-analog converters for outputs from ModeliChart. Currently, the hardware unit features 10 analog inputs and six analog outputs. The second component is a custom FMU, which can be loaded into ModeliChart. This special FMU handles the communication between ModeliChart and the hardware unit through network sockets using either a wireless or a cable connection. When loading this FMU into ModeliChart, the input and output channels of the hardware unit appear as channels which can be freely linked to channels of other FMUs. As this special FMU complies with the FMI 2.0 co-simulation standard, it can also be loaded into other FMU hosts. Therefore, our FMI-based hardware interface solution is not limited to be used in combination with ModeliChart.

An exemplary application in which we are using ModeliChart with hardware interaction are bench tests of blood pumps using a MCL. A MCL is a hydrodynamic test rig that allows applying controlled boundary conditions to a pump. This frequently includes emulation of the human cardiovascular circulation system. The hybrid MCL used in this paper (MedIT Mock, Philips Chair for Medical Information Technology, see Figure 6) consists of high-bandwidth actuators controlled by a real-time computer (dSPACE GmbH, Paderborn, Germany). It is capable of precisely enforcing arbitrary highly dynamical pressure trajectories on the pump's inlet and outlet both within the range of – 50 to 200 mm Hg. Furthermore, it provides the necessary flow to maintain the enforced pressure difference across the pump. The dynamic viscosity of blood is mimicked with a heat transfer medium for industrial plants (Glysofor N, Grafschaft, Germany). For more information on the MCL, see [14, 16]. The pump connected to this MCL is independently controlled by a dedicated custom-build console which controls the pump's rotational speed based on an analog input signal. A flow probe measures the flow through the pump resulting of the applied pressure difference and pump speed. In summary, there are three actuating variables (the inlet and outlet pressures of the pump enforced through the MCL and the pump speed) and one sensor (the flow probe). All of those actuators and sensors can be used within ModeliChart. An exemplary experiment using this setup will be presented in the section Mock circulatory loop tests of the optimal control system.

OO model-based optimal control

During a simulation within ModeliChart as shown in the section Simulation with ModeliChart, the physician is able to study the effects his inputs have on the resulting outlet flow. However, in cases where a specific output (flow) trajectory is desired, it is difficult to find the suitable necessary input signal trajectory. Moreover, especially in clinically motivated cases, it is rather of interest to only enforce single points in time or limits of several variables instead of enforcing the full trajectory over time of a single variable. Finding the necessary input signal trajectory to fulfill such requirements can be achieved with optimal control methods.

In addition to simulation applications as described in the previous sections, OO Modelica models are also well suited for optimal control applications. The objective in optimal control is to find suitable input trajectories to the system to be controlled that minimize a so called "cost function" while at the same time obeying any given restrictions on system or control variables. The cost function can be an arbitrary scalar function of the system or control variables. In general, the cost function is written as a time integral. The restrictions can be of the form of limits (for example some system variables should not become negative or stay below a maximal value) or specific values for a variable at specific times. Optimal control problems are well suited for systems featuring multiple inputs.

To illustrate the formulation of an optimal control problem, the perfusion of the arterial system is considered. The goal is to find the necessary input signal trajectory for the ideal flow source to achieve a pulsatile perfusion pressure that also fulfills various other imposed restrictions. To define the characteristics of the pulsatile perfusion pressure, the input pressure of the arterial system (here the TEW model, see Figure 2A) is coupled to the harmonic function:

$$P_{\mathrm{W,in}}(t) = A(t)\cdot\sin(2\pi\cdot t)+O(t).$$

This way the input pressure becomes sinusoidal with a frequency of 1 Hz, a time-varying amplitude $A(t)$ and a time-varying offset $O(t)$. This allows enforcing restrictions on the pulsatility characteristics. For example, the physician might propose that it should be desired to achieve a constant amplitude of 10 mm Hg. Furthermore, he might propose a minimal change of the offset over time, i.e. a minimal time derivative of $O(t)$. A suitable cost function expressing those objectives is

$$J(t)=\int_0^t q_1\cdot(A(t)-10\ \text{mm Hg})^2+q_2\cdot\dot{O}(t)^2\,dt,$$

where q_1 and q_2 are the weighting factors. They can be chosen to balance between the two goals. Furthermore, they can be used to normalize the individual terms in the cost function if their dimensions require numerical values within different orders of magnitudes. Here, we make the assumption that the amplitude goal is most important to be fulfilled and consequently choose $q_1=10^3$ and $q_2=1$.

Additionally, the physician could propose further restrictions (which are arbitrary examples here). For the experiment, a timeframe of 15 s is considered. Within this time, the physician requires the total volume $(V=\int Q_{\text{W,out}}\,dt)$ that has flown through the system to be exactly 300 ml. Moreover, he requires the volume to be exactly 75 ml at 5 s. Besides the volume, he also specifies the flow $Q_{\text{W,out}}$ to be 40 ml/s at 10 s. To enable a smooth entry and exit of the experiment, it is further required that the flow of the ideal flow source is 0 ml/s at the beginning and at the end of the experiment. Besides postulating specific values at specific times, some limits of the range of allowed numerical values of certain variables are imposed. For example, the amplitude value is limited to be positive. Furthermore, having real existing blood pump devices in mind, the allowed flow of the flow source $Q_{\text{W,in}}$ is limited to be within 0 ml/s and 60 ml/s. Additionally, to avoid negative pressures, the inlet pressure of the arterial system $P_{\text{W,in}}$ is restricted to be positive. To avoid too abrupt changes of the amplitude, the absolute value of amplitude's time derivative is limited to 10 mm Hg/s. In summary, all those specified restrictions can be written as

$$\int_0^{5\,\text{s}} Q_{\text{B}}(t)\,dt=75\ \text{ml},\ \int_0^{15\,\text{s}} Q_{\text{B}}(t)\,dt=300\ \text{ml},\ Q_{\text{W,out}}(t=10\ \text{s})=40\ \text{ml/s},$$

$$A(t=0\ \text{s})=O(t=0\ \text{s})=0\ \text{mm Hg},\ Q_{\text{W,in}}(15\ \text{s})=0\ \text{ml/s},$$

$$A(t)\geq 0,\ \dot{A}(t)\leq 10\ \text{mm Hg/s},\ 0\ \text{ml/s}\leq Q_{\text{W,in}}\leq 60\ \text{ml/s},\ P_{\text{W,in}}\geq 0\ \text{mm Hg}.$$

To consider optimization problems such as the one formulated here in combination with OO Modelica models, Optimica has been developed [2]. Optimica is a language extension to Modelica. It allows the formulation of optimization problems based on the OO principles of Modelica. As Optimica allows inheriting from existing Modelica models, it allows reusing existing code of models for the optimization. Listing 2 illustrates the implementation of the above optimization problem which inherits from the Modelica model given in Listing 2.

To find a solution to the optimization problem, different numerical methods exist. In general, the computational effort can be significantly decreased with algorithms that use the derivatives of the state variables of the model. In this regard, another advantage of OO Modelica is the fact that all equations of every component are available. Therefore, it is possible to automatically determine the derivatives through automatic differentiation techniques. The open-source software package CasADi in combination with the open-source Modelica compiler JModelica is able to automatically compile the Optimica code, to calculate the derivatives and to generate a discretized (using a direct collocation scheme) non-linear optimization problem [3, 6, 7]. The optimization problem can then be solved with an appropriate numerical solver. We use the open-source solver Ipopt [18]. The result yields the optimal trajectories for the ideal flow source that fulfill the given requirements and a minimal cost function value. A simulation with the optimal input trajectories yields the optimal pressure ($P_{\text{W,in,opt}}(t)$) and flow ($Q_{\text{W,in,opt}}(t)$) inlet trajectories for the system.

```
optimization OptimizationProblem(
   objectiveIntegrand=
      1000*(Amplitude-10)^2+der_Offset^2,
   startTime=0.0, finalTime=15.0)
 extends FlowDrivenSystem;
 input Real der_Amplitude;
 input Real der_Offset;
 Real Amplitude(start=0,fixed=true);
 Real Offset(start=0,fixed=true);
 Real y_Volume(start=0,fixed=true);
 Real y_Flow;
 constant Real pi=2*asin(1.0);
equation
 der_Offset=der(Offset);
 der_Amplitude=der(Amplitude);
 Flow_Source.cnBloodStream.P=
    Amplitude*sin(2*pi*time)+Offset;
 P_B.cnBloodStream.VolFlow=y_Flow;
 y_Flow=der(y_Volume);
constraint
 y_Volume(15.0)=300;
 y_Volume(5.0)=75;
 y_Flow(10.0)=40;
 Flow_Source.Q(15.0)=0;
 Flow_Source.cnBloodStream.P>=0;
 Amplitude>=0;
 Flow_Source.Q>=0;
 Flow_Source.Q<=60;
 der_Amplitude<=10;
end OptimizationProblem;
```

Listing 2: Optimica code of the optimization problem described in Section OO model-based optimal control.

MCL tests of the optimal control solution

Considering a real application, a common challenge is to verify that solutions obtained from model-based calculations and simulations are actually applicable to the real system. This is because many aspects of real systems are difficult to identify, to model or have simply been missed to consider. After all, models are intended to be simplified representations of real systems and finding the right amount of simplification is generally difficult and an iterative process. Therefore, within rapid control prototyping, it is becoming more and more popular to include parts of the real system into consideration early in the development and research process.

To provide an example in this regard combining all the presented tools, techniques and methods of the previous sections, a MCL experiment was performed. The aim of this experiment has been to validate whether the found solution of the optimal control problem can be carried out in reality if no ideal flow source exists. Therefore, the ideal flow source is to be exchanged for a real existing blood

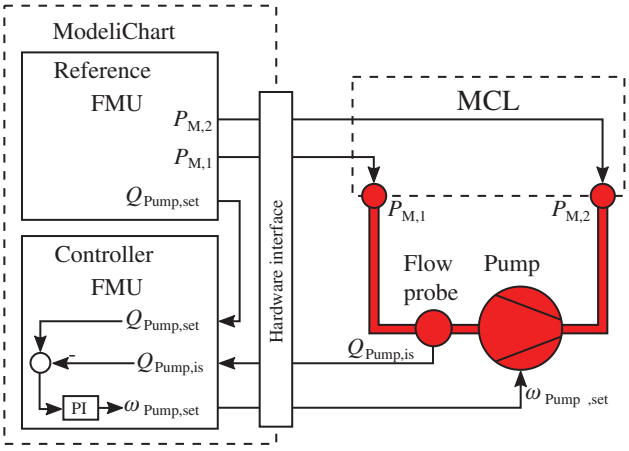

Figure 7: Concept of the experimental setup described in the section Mock circulatory loop tests of the optimal control solution.

pump. In the experiment, a deltastream DP3 blood pump (Medos, Stolberg, Germany) was employed.

The DP3 is diagonal pump featuring a magnetically mounted impeller driven by a brushless electric motor. A custom-build controller allows controlling the rotational speed of the impeller according to an analog voltage input signal. The actual flow through the pump mainly depends on the rotational speed of the impeller and on the pressure difference across the pump. Although the flow dynamics of the pump could be modeled in principle, the pump is considered to be a "black box" of unknown dynamics here.

To reproduce the calculated optimal pressure and volume trajectories at the outlet of the ideal flow source, a setup according to Figure 7 was utilized. The MCL was used to impress the pressure difference on the DP3. It was assumed that the DP3 is able to draw volume from a reservoir with constant pressure (here chosen to be 0 mm Hg). Accordingly, the pressures of the MCL are set to be $P_{M,1}(t) = 0$ mm Hg and $P_{M,2}(t) = P_{W,in,opt}(t)$. The speed of the DP3 is controlled using a proportional integral (PI)-controller $(y = K \cdot (u + 1/T_i \cdot \int u\, dt))$ aiming at generating the calculated optimal flow trajectory $(Q_{pump}(t) \stackrel{!}{=} Q_{W,in,opt}(t))$.

This setup has been realized with ModeliChart interfacing the MCL and the pump using two FMUs plus the hardware interface FMU (Figure 7). The controller FMU implements the PI-controller. The reference FMU plays back the calculated optimal pressure and flow trajectories. The conversion of the unit values into the corresponding analog voltage values is performed within the channel links.

Results

As an illustrative example, a small MiL simulation without hardware interaction performed in ModeliChart can be seen in Figure 3. For examples and results of larger clinically motivated simulation *in silico* studies that have been conducted with ModeliChart, see the articles of Habigt et al. [12] and Moza et al. [17] within this special issue.

The results of the optimization problem are given in Figure 8. As can be seen in the plots, all the specified requirements are fulfilled by the solution found. The results of the MCL experiment using a real pump are provided in Figure 9. Suitable parameters for the PI-controller were roughly obtained based on an iterative trial-and-refine procedure. The parameters that have led to the results in Figure 9 are $K = 45$ and $T_i = 0.0075$ s. The experiment yields the rotational speed trajectories of the pump. As the real pump is only capable of positive rotational speeds, it stops when the control algorithm demands negative rotational speeds (Figure 9, bottom).

Discussion

Within the scope of the exemplary case, the obtained results enable the physician and the engineer to conduct a trial on a real systemic circulation system – possibly within an animal trial (*in vivo*) or an artificial mockup of the systemic circulation (*in vitro*). Based on the results, they might find that the model used (i.e. the TEW for the arterial system) is not an adequate description of the real system. Typically, deviations between the model and the real system can be due to structural deficiencies of the model or due to wrong model parameters. In both cases, the principles of OO modeling in combination with ModeliChart and the methods described in the previous sections can be used to refine the model.

When considering the model structure improvements, a major benefit of OO Modelica models is that individual components can easily be exchanged with minimal effort. Although new components could alter the original cause and effect chain (such as the block diagrams within Figure 2), the new resulting system equations are automatically generated by the Modelica compiler. In contrast, using a signal-based modeling or a state-space modeling approach, new components can result in significant effort to implement the changes required. Furthermore, Modelica offers sophisticated mechanisms that allow reusing a great amount of code. An example of this is the already mentioned replaceable keyword which has been used to mark the arterial system component in the models in Listing 1. The constrainedby keyword specifies that the component can only be replaced by models inheriting from the specified model to ensure compatibilities of the connectors. Listing 3 shows how the old TEW component can be replaced by a new arterial system component using the redeclare keyword. This allows to reuse the code of the existing flow-driven system model. Similarly, the Optimica

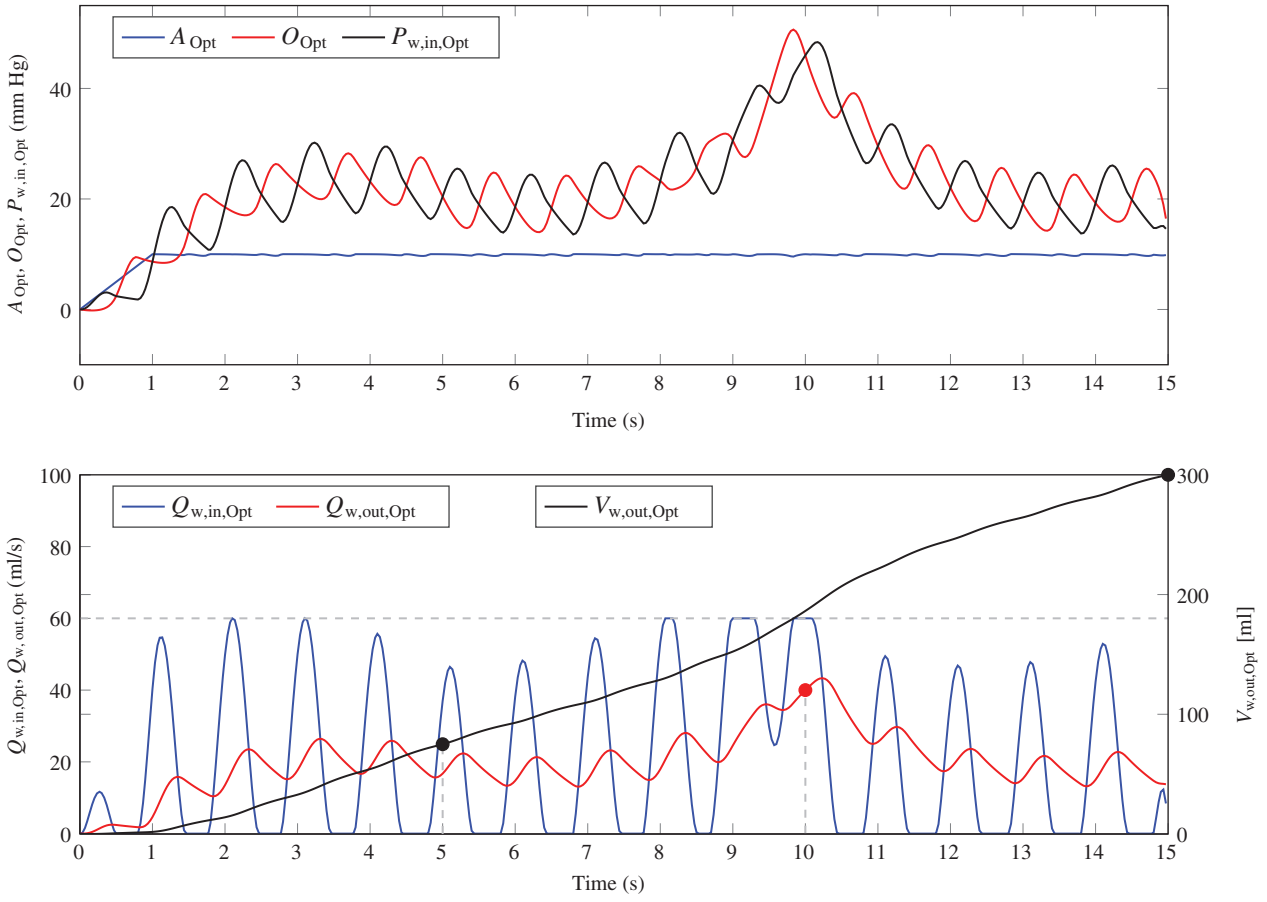

Figure 8: Results of the optimization problem described in the section Object-oriented model-based optimal control. The imposed restrictions on variables are marked in the plot.

code of the optimization problem can be reused by only replacing line 5 in Listing 2 with the code of lines 11–13 of Listing 3. All in all, those mechanisms of OO modeling enable the engineer and the physician to quickly evaluate new structures of their models.

Considering refinement of model parameters, a rough tuning of the model parameters can be quickly done within a simulation in ModeliChart. Although manually tuning the parameters only leads to approximations of their ideal values, we have made the experience that this process conveys a feeling of the influence and sensitivity of the individual parameters to the user. This forms an ideal base for a possible consecutive numerical parameter optimization procedure. There exist various toolboxes to perform such numerical parameter optimization procedures within many Modelica development environments. One possible method for performing such a parameter optimization is to rewrite the parameter optimization problem as an optimal control problem. For this, the parameters of interest are introduced as inputs to the models. The cost function is chosen in a way that any differences between the measurements and the simulation values are penalized. For constant parameters, additional restrictions on the derivatives of the parameters can be enforced. A so-formulated optimal control problem can then be solved as described in the section Object-oriented model-based optimal control. An application of this procedure can be found in the corresponding article by Moza et al. within this special issue [17].

Considering the presented optimal control method, a general limitation is the controller being only feed-forward, i.e. no feedback during the application is considered. In case of unconsidered noise or severe model inaccuracies, optimal control might lead to unsatisfactory results. Therefore, it can be indicated to combine optimal control with other control strategies to account for those issues. Nevertheless, optimal control allows for the realization of clinically motivated goals. Examples which use optimal control in combination with the technically assisted cardiovascular systems can be found in [5, 11]. A popular extension of optimal control is the concept of a model-based predictive controller (MPC). In an MPC, the

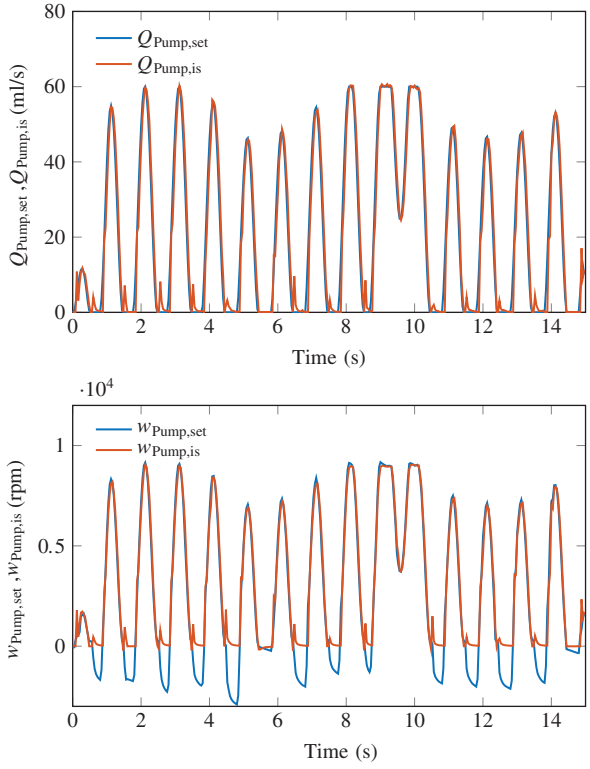

Figure 9: Results of the MCL experiment described in the section Mock circulatory loop tests of the optimal control solution.

```
model NewArterialSystem
 import HumanLib.Basics.*;
 extends Interfaces.BloodStreamTwoPin;
 parameter Real p1;
  ...
 equation
  ...
end NewArterialSystem;

model NewFlowDrivenSystem
 extends FlowDrivenSystem(
    redeclare NewArterialSystem
       ArterialSystem(p1=1));
end NewFlowDrivenSystem;
```

Listing 3: Modelica code demonstrating replacing the arterial system component.

optimization is performed during runtime and periodically repeated based on the current actual system state. However, designing an MPC imposes requirements on the computation time of the optimization and generally requires significant conceptual and implementation effort and expertise dependent on the application. Thus, refining the model is accompanied with significant effort to adapt the MPC.

As it has been demonstrated within the above examples, there are various SiL and HiL setups that can be realized with ModeliChart in combination with the hardware interface. Special use cases include signal generation of arbitrary signals (only limited by the used sample rate) and data acquisition and recording. Considering the presented MCL, in a common operation mode for left heart ventricular assist devices (VAD) (pumps that are intended for not replacing but supporting the heart, VAD) tests, a model of the cardiovascular system is simulated in ModeliChart. The simulated left ventricular and aortic pressures are set as inlet and outlet pressures of the device under test. The measured flow through the connected VAD is fed back to the cardiovascular system simulation as additional flow from the left ventricle to the aortic arch. Consequently, the simulated pressures adapt to the actual flow through the VAD. The setup allows for a quick change of the used cardiovascular model FMU during runtime without the need to recompile anything. As the current focus of development of the HumanLib is on the diseased cardiovascular system, a common VAD controller trial includes testing the behavior of a controller for different pathological conditions of the cardiovascular system. Another aspect of VAD research focuses on the question of suitable control objectives to allow safe and efficient treatment (for a comprehensive overview see [4], the objectives that have been considered in the preceding sections are hardly clinically applicable). Similarly, the setup allows to quickly compare different VAD controller strategies.

Considering the performance of ModeliChart, it needs to be pointed out that the current software and hardware architectures is not able to fulfill hard real-time requirements, as the time needed for the execution of a single simulation step and update of the hardware interface is technically not deterministic. This is due to the fact that Windows® is not a real-time operating system which might lead to ModeliChart not being allotted the necessary computational resources in time. Furthermore, the communication with the hardware interface is not deterministic as the current implementation through network sockets is based on an own protocol on top of the user datagram protocol (UDP). Besides, if the communication failed, the hardware interface would not be able to receive new values and would hold the last-received value. Having said that, in practice those shortcomings hardly impair the eligibility of ModeliChart; using an ordinary contemporary computer, sample times within the order of single milliseconds are achievable (e.g. a sample time of 2 ms has been used for the experiment in the section Mock circulatory loop tests of the optimal control solution). Nevertheless, we are currently working toward an alternative architecture featuring the simulation engine running on a real-time-operating system, preferably on the same system that possesses the analog interfacing hardware.

The overall design goal of ModeliChart has been to provide a tool tailor-made to the requirements of the parties involved within the field of biomedical engineering. The ongoing development of ModeliChart is supposed to be based on an open innovation process; we highly encourage other groups that find themselves with different scenarios and use cases as described in this paper to contact us to evaluate ModeliChart. As mentioned, ModeliChart as well as the necessary additional software to build a hardware interface will be made available free of charge upon request.

Acknowledgments: This work was supported by the German Research Foundation (DFG) within the Smart Life Suport 2.0 PathoMod project (PAK 183-2).

References

[1] Abel D, Bollig A. Rapid control prototyping. Berlin, Heidelberg: Springer 2006.
[2] Åkesson J. Optimica – an extension of modelica supporting dynamic optimization. In: Proceedings of the 6th International Modelica Conference 2008. Bielefeld, Germany 2008.
[3] Åkesson J, Årzén K-E, Gäfvert M, Bergdahl T, Tummescheit H. Modeling and optimization with Optimica and JModelica.org – Languages and tools for solving large-scale dynamic optimization problems. Comput Chem Eng 2010; 34: 1737–1749.
[4] AlOmari A-HH, Savkin AV, Stevens M, et al. Developments in control systems for rotary left ventricular assist devices for heart failure patients: a review. Physiol meas 2012; 34: R1.
[5] Amacher R, Asprion J, Ochsner G, et al. Numerical optimal control of turbo dynamic ventricular assist devices. Bioengineering 2013; 1: 22–46.
[6] Andersson J, Åkesson J, Casella F, Diehl M. Integration of CasADi and JModelica.org. In: Proceedings of the 8th International Modelica Conference. Dresden, Germany 2011: 218–231.
[7] Andersson J, Åkesson J, Diehl M. CasADi: a symbolic package for automatic differentiation and optimal control. Berlin, Heidelberg, Germany: Springer 2012.
[8] Blochwitz T, Otter M, Åkesson J, Arnold M, Clauss C, Elmqvist H, et al. Functional mockup interface 2.0: the standard for tool independent exchange of simulation models. In: Proceedings of the 9th International Modelica Conference. Munich, Germany: The Modelica Association 2012: 173–184.
[9] Brunberg A, Maschuw J, Autschbach R, Abel D. Object-oriented model library of the cardiovascular system including physiological control loops. In: Proceedings of the World Congress on Medical Physics and Biomedical Engineering, September 7–12, 2009, Munich, Germany. Springer 2009: 166–169.
[10] Fritzson P. Principles of object-oriented modeling and simulation with Modelica 2.1. Hoboken, New Jersey: John Wiley & Sons 2010.
[11] Gesenhues J, Hein M, Habigt M, Mechelinck M, Albin T, Abel D. Nonlinear object-oriented modeling based optimal control of the heart: performing precise preload manipulation maneuvers using a ventricular assist device. In: Proceedings of the 2016 European Control Conference (ECC). 2016: 2108–2114.
[12] Habigt M, Ketelhut M, Gesenhues J, Schrödel F, Hein M, Mechelinck M, et al. Comparison of novel physiological load-adaptive control strategies for ventricular assist devices. Biomedical Engineering/Biomedizinische Technik 2016. doi: https://doi.org/10.1515/bmt-2016-0073. [Epub ahead of print].
[13] Heinke S, Pereira C, Leonhardt S, Walter M. Modeling a healthy and a person with heart failure conditions using the object-oriented modeling environment Dymola. Med Biol Eng Comput 2015; 53: 1049–1068.
[14] Heinke S, Schwandtner S, Costante C, et al. Modellierung und Regelung eines hydraulischen HIL-Simulators zum Test von Herzunterstützungssystemen/Modeling and Control of a Hydraulic Simulator for Ventricular Assist Device Testing. at-Automatisierungstechnik 2013; 61: 645–655.
[15] Mateják M, Kulhánek T, Šilar J, Privitzer P, Jezek F, Kofránek J. Physiolibrary-Modelica library for physiology. In: Proceedings of the 10th International Modelica Conference; March 10–12; 2014; Lund; Sweden: Linköping University Electronic Press 2014: 499–505.
[16] Misgeld BJE, Rüschen D, Schwandtner S, Heinke S, Walter M, Leonhardt S. Robust decentralised control of a hydrodynamic human circulatory system simulator. Biomed Signal Proces Control 2020; 15: 35–44.
[17] Moza A, Gesenhues J, Autschbach R, et al. Parametrization of an in-silico circulatory simulation by clinical datasets – towards prediction of ventricular function following assist device implantation. [Epub ahead of print].
[18] Wächter A, Biegler LT. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math Program 2006; 106: 25–57.
[19] Westerhof N, Lankhaar J-W, Westerhof BE. The arterial windkessel. Med Biol Eng Comput 2009; 47: 131–141.

Supplemental Material: The online version of this article (DOI: 10.1515/bmt-2016-0074) offers supplementary material, available to authorized users.