SIMULATION-BASED DEVELOPMENT AND OPERATION OF CONTROLS ON THE BASIS OF THE DEVS FORMALISM[1]

Tobias Schwatinski[1], Thorsten Pawletta[1], Sven Pawletta[1], Christian Kaiser[2]

[1]Hochschule Wismar – University of Applied Sciences: Technology, Business and Design
Research Group Computational Engineering and Automation
[2]TLK-Thermo GmbH Braunschweig

{tobias.schwatinski, thorsten.pawletta, sven.pawletta} @ hs-wismar.de

Abstract

The Discrete Event System (DEVS) Formalism is based on systems theory and provides an extensive framework for modeling and simulation of discrete event and hybrid systems. This paper investigates the suitability of Parallel DEVS (PDEVS), a general DEVS extension, and of the specific Real-Time DEVS (RT-DEVS) extension for a throughout simulation-based development of discrete event controls. The research is based on the V-Model that generally describes the control development process, whereas main focus is set to the Rapid Control Prototyping (RCP) concept. As a result the PDEVS and RT-DEVS specifications are integrated to form an extended DEVS specification called PDEVS-RCP. Finally, the usage and functionality of PDEVS-RCP is demonstrated using a robot control application.

Keywords: discrete event simulation, discrete event control, DEVS, RCP, robot controls.

Presenting Author's biograph

Tobias Schwatinski has studied mechanical engineering at Wismar University and received his Master degree in January 2010. Currently he is working on new approaches in the field of cooperative robot controls.

[1] This work is supported by the German Federal Ministry of Education and Research (support code 1747X08).

1 Introduction

Following [1], all steps of the development process for complex controls in the field of automation are summarized in the V-Model:

• Conceptual formulation, requirement specification, functional specification

• Analyzing and modeling of technical processes

• Development of control algorithms

• Coding and implementation of algorithms on destination hardware

• Successive placing into operation of controls

• Bringing into service of the complete control

Today a large number of software environments exist for nearly all steps in the V-Model. In particular the interfaces between different software environments or rather the users of this software are important weak points in the whole development process. Often different software environments are not compatible among each other. In addition, a loss or a misinterpretation of information can take place during the information exchange between several users. Reasons may be different methods or the large amount of different software tools used in the development process at all.

The objective of the *Rapid Control Prototyping* (RCP) approach is to shorten, to simplify and to reduce the error probability of the entire development process. Hence, a continuous usage of compatible software tools and model-based development methods based on a well defined theoretical background are central elements of the RCP approach, particularly to avoid unnecessary re-implementations. The continuous usage of compatible software tools from early planning-phases till placing into operation is called Tool-Chain. A common representative of such a Tool-Chain is the scientific and technical computational environment Matlab/Simulink with its additional toolboxes. Matlab/Simulink offers several simulation tools for system and control design and tools for automatic code generation for specific programmable controllers. An alternative approach to the usage of specific programmable controllers is the so called *Software in the Loop* (SiL) concept. Here conventional PCs or industry PCs are used during the operation phase. For this purpose appropriate process-interfaces have to be implemented. This specific kind of communication with real processes is also called *implicit code generation*. One specific type of the SiL concept is the *Simulation-based control approach* (SBC) according to [3, 4, 5]. It is based on simulation models which are stepwise enhanced during the design and automation phase to a full control system using the implicit code generation approach. Moreover, the SBC approach allows performing additional state calculations and process optimizations during real-time operation, because the control system contains a detailed process model [3].

Up to now research to the Simulation-based control approach and to the RCP concept is mostly based on the conventional Matlab/Simulink tools. This paper investigates the Simulation-based control approach following the Discrete Event System (DEVS) formalism introduced by Zeigler [6] and its extensions Parallel DEVS (PDEVS) [7] and Real-Time DEVS (RT-DEVS) [2, 6]. A main goal is to reinforce the theoretical background of the Simulation-based control approach. Therefore the DEVS formalism is shortly introduced on the basis of the Parallel DEVS (PDEVS) specification and important extensions of RT-DEVS are discussed and analyzed with regard to general requirements of the RCP concept. Then the paper gives detailed proposals how PDEVS and RT-DEVS can be integrated to fulfill the RCP requirements using the Simulation-based control approach. Finally, the new ideas are illustrated by a robot control application.

2 Specification and dynamic behavior of Parallel DEVS

The *Discrete EVent System* (DEVS) formalism was introduced by Zeigler in 1976. It provides a comprehensive framework for modeling and simulation based on systems theory. The formalism includes on the one hand detailed model specifications and on the other hand corresponding simulation algorithms. Besides, modeling is based on a modular, hierarchical specification. The dynamic behavior is specified in *atomic DEVS* systems. Furthermore, there are *coupled DEVS* systems, which are used for composing atomic DEVS and coupled DEVS respectively. Every coupled DEVS can be a part of any other coupled DEVS. Following the classic DEVS approach coupled DEVS systems specify only a system structure and do not contain a separate dynamic description. For the simulation phase a *simulator* is assigned to any atomic DEVS and a *coordinator* to any coupled DEVS as execution controller. Moreover, there is one *root coordinator* above all simulators and coordinators, which controls the outermost coordinator and increases the simulation time until any stop criterion will be reached. The whole simulation process is event-based and all simulators and coordinators communicate during simulation phase with several messages with each other.

The DEVS formalism has been extended and adapted for different purposes. An important modification is the *Parallel DEVS* (PDEVS) approach from Chow [7]. This one eased the specification of simultaneous events and provides a real concurrent execution of DEVS models by modifying the classical simulator and coordinator algorithms.

A Parallel atomic DEVS is specified by [6] as follows:

$$PDEVS_{atomic} = \{X, Y, S, \delta_{ext}, \delta_{int}, \delta_{conf}, \lambda, ta\}$$

where X is the set of input values, S is the set of sequential states, Y is the set of output values, δ_{ext} is the external transition function, δ_{int} is the internal transition function, δ_{conf} is the confluent function used for simultaneous external and internal events, λ is the output function and *ta* is the time advance function used for calculating the logical time advance. The dynamic behavior of a Parallel atomic DEVS is shown in Fig. 1 (a) and (b).

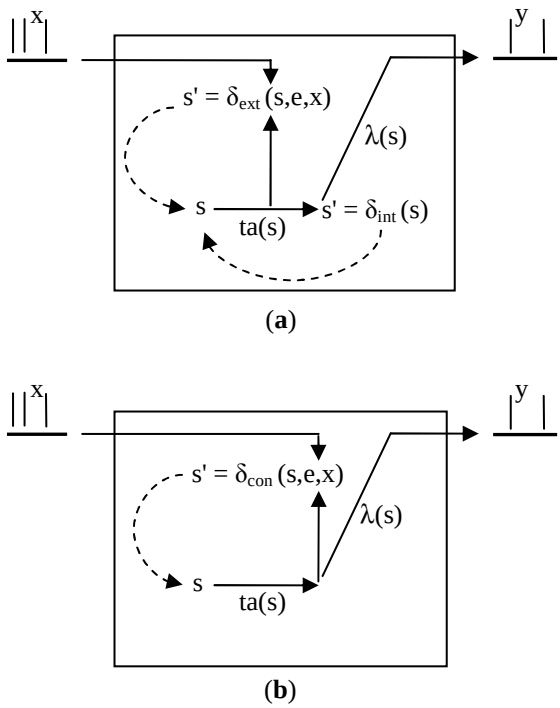

Fig. 1 Dynamic behavior of a Parallel atomic DEVS according to [6]

Following Fig. 1 (a) every atomic DEVS has an internal state $s \in S$. By the mean of the time advance function *ta(s)* the time step till the next internal event is calculated on the basis of the internal state *s*. After having expired this time period the output function $\lambda(s)$ is carried out and as a result all output events $y \in Y$ at the output port are calculated on the basis of the internal state *s*. In the following the internal transition function $\delta_{int}(s)$ is carried out, which calculates the next state $s' \in S$ on the basis of the current state *s*. If external events $x \in X$ occur at the input port the external transition function $\delta_{ext}(s,e,x)$ will be executed. This function calculates the next state $s' \in S$ on the basis of the current state *s* and the elapsed time *e* since the last event and the current external events $x \in X$. At the end the logical time advance for the next internal event is calculated by the time advance function *ta(s)*.

Fig. 2 (b) shows the dynamic behavior of a Parallel atomic DEVS if internal and external events occur simultaneously. At first output events $y \in Y$ at the output port are calculated by the output function $\lambda(s)$ and at second the next internal state $s' \in S$ is calculated by the confluent function $\delta_{con}(s,e,x)$. The processing of simultaneous events is handled with the confluent function δ_{con} by any atomic DEVS on its own. That's why all atomic DEVS are able to operate concurrently and independently from each other.

A Parallel coupled DEVS is specified by [6] as follows:

$$PDEVS_{coupled} = \{X, Y, D, \{M_d \mid d \in D\}, Z_{i,d}\}$$

where X is the set of input values, Y is the set of output values, D is the set of the component names, M_d is the DEVS system of component name $d \in D$ of the coupled DEVS. $Z_{i,d}$ defines the coupling relations of internal components with each other or to any output / input ports of the coupled DEVS based on output / input relations $(i \rightarrow d)$.

3 Specification and dynamic behavior of Real-Time DEVS

Real-Time DEVS (RT-DEVS) extends the classic DEVS Theory. It is introduced for DEVS models, which should be simulated in real-time and it supports interaction with a hardware environment. Following [2] every RT-DEVS model is directly simulated in real-time by an appropriate *real-time-simulator*. An atomic RT-DEVS system is specified by [6] as follows:

$$RT\text{-}DEVS_{atomic} = \{X, S, Y, \delta_{ext}, \delta_{int}, \lambda, ti, \psi, A\}$$

Here *X*, *S*, *Y*, δ_{int} and λ have the same definition as for Parallel atomic DEVS. In contrast to Parallel DEVS the definition of the external transition function δ_{ext} is slightly modified[2]. In addition, a set of executable activities *A* including any constrains, a time interval function *ti* and an activity mapping function ψ are defined. The dynamic behavior of an atomic RT-DEVS is shown in Fig. 2.

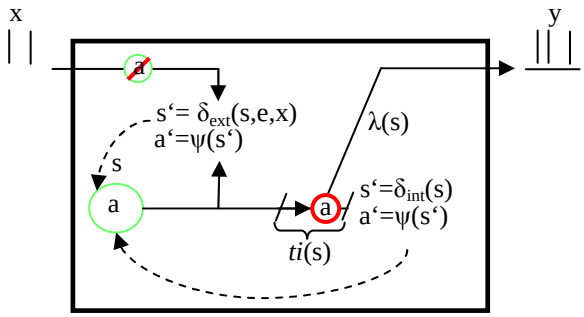

Fig. 2 Dynamic behavior of an atomic RT-DEVS

[2] $PDEVS_{atomic}$ δext: Q×X → S with Q={(s,e) | s∈S, 0 ≤ e ≤ ta(s)}
$RT\text{-}DEVS_{atomic}$ δext: Q×X → S with Q={(s,e) | s∈S, 0 ≤ e ≤ $ti(s)_{max}$}
with $ti(s)_{max}$ as a maximum of the execution time of an activity a∈A

Every state $s \in S$ is related by the activity mapping function $\psi(s)$ with one activity $a \in A$ and with a minimal and maximal execution time by the time interval function *ti(s)*. Each activity *a* can be seen as an executable function that is not allowed to receive or send events or to change internal states. The minimal and maximal execution time is calculated by $ti(s)=[comp\_time - \varepsilon, comp\_time + \varepsilon]$ and provides upper and lower bounds within an activity *a* should be completely processed. In this context ε is the allowable tolerance of an estimated execution time *comp_time*. An internal event occurs if a currently executed activity *a* ends within the time interval *ti(s)*. In analogy to the behavior of a Parallel atomic DEVS at first $\lambda(s)$ is carried out and all output events $y \in Y$ at the output port are calculated and at second the internal transition function $\delta_{int}(s)$ is carried out, which calculates the next state $s' \in S$. After that the new activity $a' \in A$ and the new time interval *ti(s')* are calculated. If external events $x \in X$ occur at the input port at first the currently executed activity *a* is aborted and at second the next state $s' \in S$ is calculated by the external transition function $\delta_{ext}(s,e,x)$. Moreover, the new activity $a' \in A$ is calculated by the activity mapping function $\psi(s')$ and the new time interval *ti(s')* is determined.

The specification of coupled DEVS is the same as for Parallel DEVS. In the simulation phase a *simulator* is assigned to any atomic RT-DEVS and a *coordinator* to any coupled DEVS as execution controller. But this time the time advance during simulation is based on *real-time* and not on a *logical time*. Especially this fact and the described differences in the system specification and the dynamic behavior lead to simulator- and coordinator algorithms that differ strongly from Parallel DEVS.

4 Analyzing the suitability of both DEVS formalisms concerning the RCP concept

Following Abel [1] the most important aspects of the *Rapid Control Prototyping* (RCP) approach are the continuous and model-based development of applications using a compatible Tool-Chain. Within the design phase any technical process is modeled in a so called *process model*. On the basis of these process models different control algorithms can be implemented as *control models* and tested using simulation. In the next step selected control models should be transformed without manual re-implementation to real control software. This code transformation can be realized by an *explicit code generation* for specific target platforms using specific compilers or by an *implicit code generation* following the Software in the Loop (SiL) concept using appropriate process interfaces. In the SiL approach development PCs or any common industrial PCs are directly used as control hardware. A specific type of SiL is the *Simulation-based control approach* by [3, 4, 5]. Here, process models and selected control models from the design phase are stepwise enhanced by using the implicit code generation and finally they are used as control software for the real process. On the one hand this procedure increases the safety of the entire system and on the other hand it reduces the costs of development because of avoiding manual re-implementations. Moreover, the integration of process models into the control phase allows additional state calculations, increases the quality of the control application and it could reduce hardware costs by e.g. using less sensors.

Fig. 3 shows schematically a comparison of the general Simulation-based control (SBC) approach and its implementation using the former introduced DEVS formalisms.

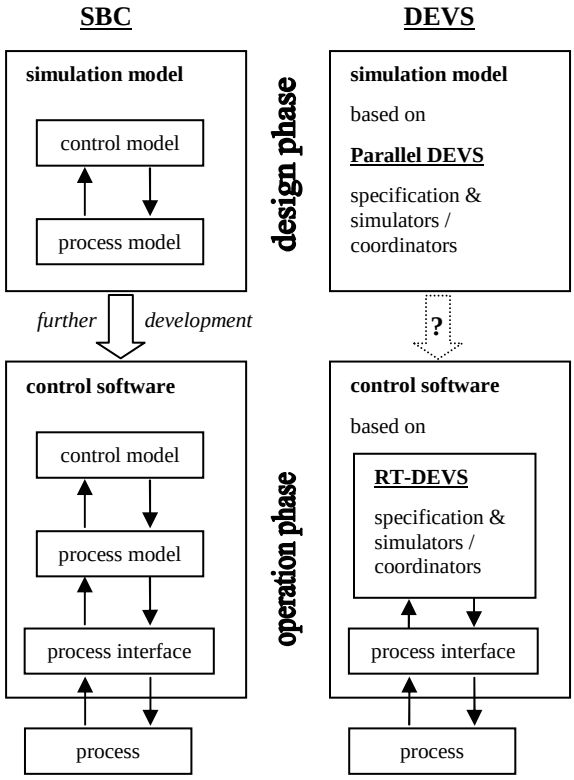

Fig. 3 Comparison of the general Simulation-based control (SBC) approach and its implementation with DEVS formalisms

On the one hand Parallel DEVS fulfills all requirements for simulation models in design phase, whereas the requirements of control software in operation phase are fulfilled by the RT-DEVS approach. However, there is no way to successively enhance PDEVS models from design phase to RT-DEVS models for operation phase, because their atomic system specifications and their simulation algorithms strongly differ from each other.

5 Integration of PDEVS and RT-DEVS to PDEVS-RCP

A useful approach to overcome the previously discussed deficits is to integrate both DEVS formalisms with the aim to fulfill the requirements of the design and operation phase respectively. In the following such integration is described by extending the Parallel DEVS formalism. The extension is called PDEVS-RCP. An atomic PDEVS-RCP system is specified as follows:

$PDEVS\text{-}RCP_{atomic} = \{X, S, Y, A, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda, ta\}$

$S, Y, \delta_{ext}, \delta_{int}, \delta_{con}, ta$ are analog to atomic PDEVS

A set of executable activities

$X = \{X_{model}, X_{clock}\}$ set of input events

with X_{model} set of system specific input events

X_{clock} set of real-time values of an extern Real-Time Clock (RTC)

$\lambda: \; S \rightarrow Y \times A$ combined output and activity mapping function

Thereby S, Y, δ_{ext}, δ_{int}, δ_{con} and ta are defined in accordance with PDEVS. Following RT-DEVS, A contains the set of executable activities. The set of input events X consists on the one hand of system specific input events X_{model} that correspond to the set X of an atomic PDEVS or an RT-DEVS respectively and on the other hand of real-time values X_{clock}. The latter are generated by a real-time clock (*RTC*) that takes influence on any PDEVS-RCP by external events. The combined output and activity mapping function λ defines analog PDEVS the generation of output events $y \in Y$. In addition, this function defines similar to a DEVS extension in [8] the state based mapping of activities $a \in A$ according to the activity mapping function ψ of RT-DEVS specification. In contrast to RT-DEVS there exists no special function for the calculation of minimal and maximal execution times of activities according to the time interval function *ti* of RT-DEVS.

The dynamic behavior of an atomic PDEVS-RCP under real-time conditions is shown in Fig. 4. The execution of simultaneous external and internal events using the confluent function δ_{con} is not shown in figure 4, because it is identical to atomic PDEVS. However, the real-time execution of an atomic PDEVS-RCP is based on the following conditions.

- The state based minimal and maximal time interval $ti = [comp\_time - \varepsilon, comp\_time + \varepsilon]$ of an activity $a \in A$ has to be explicitly saved within the system state as real-time values. The update of *ti* has to be performed by the transition functions δ_{int}, δ_{ext} or by the confluent function δ_{con}.
- An external real-time clock (RTC) component, implemented as an ordinary atomic PDEVS, has to send the real-time clock values as external events $x_{clock} \in X_{clock}$ to any atomic PDEVS-RCP, which stores the current real-time clock value as state value using its external transition function δ_{ext}. The time interval between two x_{clock} events has to be smaller or equal to the smallest logical time step *min(ta(s))* of the whole model. From this it follows that any logical time step *ta(s)* till the next internal event of an atomic PDEVS-RCP can be either *0* or ∞.

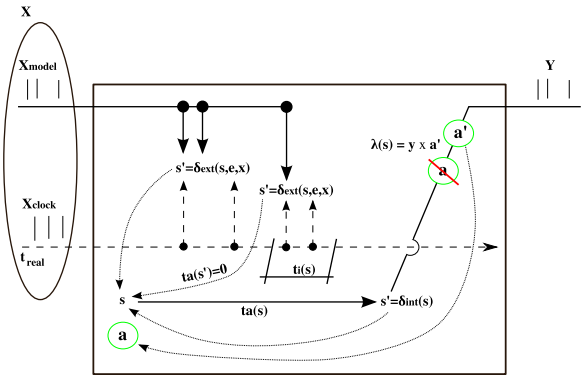

Fig. 4 Dynamic behavior of an atomic PDEVS-RCP[3]

The basic dynamic behavior of an atomic PDEVS-RCP is the same as of ordinary atomic PDEVS. Differences result from the introduced modeling extensions. Every atomic PDEVS-RCP has an internal state $s \in S$ that is related to an activity $a \in A$. The minimal and maximal execution time $[ti_{min}, ti_{max}]$ of the currently running activity *a* are saved as real-time values in the state *s*. The time advance function *ta(s)* calculates the logical time step till the next internal event according to the former described second condition. If the internal event occurs, at first the output function $\lambda(s)$ is carried out. It generates on the basis of the internal state *s* the output events $y \in Y$ at the output port, cancels the currently running activity *a* and initializes the next activity *a'*. At second, the internal transition function $\delta_{int}(s)$ is carried out. It calculates on the basis of the internal state *s* the next state *s'* including the real-time values for the execution time boundaries $[ti_{min}, ti_{max}]$ of the new activity *a'*. At third the logical time step *ta(s')* till the next internal event is calculated.

External events are subdivided in events from system specific DEVS systems $x_{model} \in X_{model}$ and events from the real-time clock $x_{clock} \in X_{clock}$. If external events $x \in X$ occur the external transition function $\delta_{ext}(s,e,x)$ is carried out. This function calculates on the basis of the internal state *s*, the elapsed time *e* since the last event

[3] The identifier *ti(s)* in Fig. 4 is not a time interval function. It is a state vector $[ti_{min}, ti_{max}]$, which contains the execution boundaries in real time for the currently running activity *a*.

and all input events *x* the next state *s'*. In addition, $\delta_{ext}(s,e,x)$ checks on the basis of the real-time values saved in the internal state *s* if any external event takes place within the time interval $[ti_{min}, ti_{max}]$.

- If external events occur outside the interval $[ti_{min}, ti_{max}]$ the next internal state *s'* is computed including recalculated $[ti_{min}, ti_{max}]$ state values. Moreover, the logical time advance *ta(s')* is calculated.
- If external events occur within the interval $[ti_{min}, ti_{max}]$ the next internal state *s'* is computed. The next state *s'* has to lead to *ta(s')=0* and provokes immediately an internal event without a further logical time step.

The integration of real-time clock values by means of external events and the storage of a real-time interval *ti* in the system state are central elements of PDEVS-RCP. Atomic PDEVS-RCP systems can be integrated in ordinary coupled DEVS as defined in the PDEVS formalism. Moreover, the execution algorithms of a simulator, coordinator and root-coordinator defined in the PDEVS formalism remain unchanged. That's why any model including atomic PDEVS-RCP systems can be executed using the ordinary PDEVS algorithms within the design phase and the operation phase.

6 Example: A Robot control with DEVS

This section discusses an example of a robot control on the basis of the Simulation-based control approach using the PDEVS-RCP extension. Fig. 5 shows the control structure of a robot taken from [9], which is able to perform asynchronous point to point (PTP) movements and to stop at any point in the workspace for a specified time period.

The coupled DEVS *RobotControl* consists of two real-time depending atomic PDEVS-RCP systems called *Control* and *PInterface* as well as two atomic PDEVS systems called *Process* and *RTC*. The component *Control* defines the real-time based control logic. *Process* specifies a process model of the robot and stores for example joint and gripper states. *PInterface* defines an "implicit" process interface to the robot controller that directly accesses the robot actors and

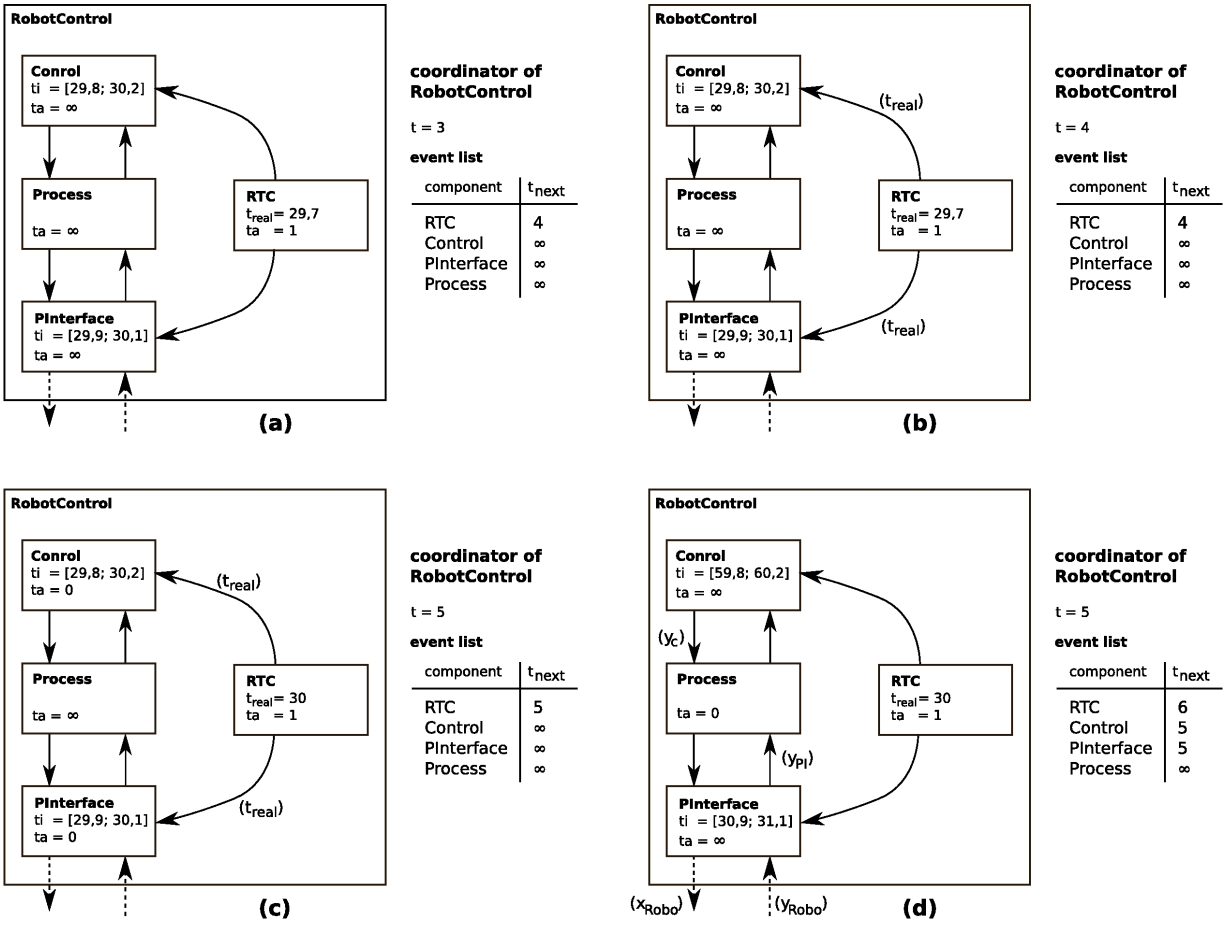

Fig. 5 Control structure of a robot on the basis of the Simulation-based control approach using PDEVS-RCP

sensors. *RTC* acts as an interface to the real-time clock and sends real-time values as output events periodically.

In the following, the operation of the control structure pictured in figure 5 is briefly described. Each subfigure pictures the logical time (simulation time) *t* and the event list of the topmost coupled DEVS *RobotControl*. For this example we assume that an internal event occurs for *Control* every $30 \pm 0{,}2\ s$. Furthermore, an internal event occurs for *PInterface* every $1 \pm 0{,}1\ s$, because control commands have to be sent to the execution controller of the robot and sensor values have to be received within function $\lambda(s)$ periodically. *RTC* schedules its next internal event periodically with the logical time step $ta=1$ and sends the real-time t_{real} as event to the real-time depending components *Control* and *PInterface* by $\lambda(s)$. Fig. 5 (a) shows the event list of the coordinator at the simulation time $t=3$ and at the real-time $t_{real}=29.7$. The next internal event occurs for RTC at the logical time $t=4$. According to Fig 5 (b) the simulation time is set to $t=4$ and RTC sends $t_{real}=29.7$ as event to *Control* and *PInterface*. Both components check using their external transition function δ_{ext} whether the transmitted real-time is within the time interval *ti* or not. In fact t_{real} is not within *ti* and that's why for both components $ta=\infty$ is set and no internal event has been scheduled. *RTC* schedules its next internal event with $ta=1$. Hereupon, the next internal event occurs for *RTC* at logical time $t=5$. According to Fig 5 (c) the simulation time is set to $t=5$ and RTC sends $t_{real}=30$ as event to *Control* and *PInterface*. Both components check again using their external transition function δ_{ext} if $t_{real}=30$ is inside their time interval *ti* - in fact this is the case. Thus, *Control* and *PInterface* schedule their next internal events with $ta=0$ at $t=5$ whereas *RTC* schedules its next internal event with $ta=1$ at $t=6$. Following Fig. 5 (d) an internal event take place for *Control* and *PInterface* at simulation time $t=5$. At first the λ-function of *Control* and *PInterface* is carried out and the output events (y_C) and (y_{PI}) are sent to *Process*. In addition, *PInterface* communicates with the robot execution controller by sending x_{Robo} and receiving y_{Robo}. At second both components carry out their internal transition function δ_{int} at $t=5$ and calculate amongst others their new time interval *ti*. At third *PInterface* and *Control* calculate their next internal event time advance with $ta=\infty$. Concurrent *Process* receives its external events (y_C) and (y_{PI}) at simulation time $t=5$ and calculates using the external transition function δ_{ext} its next internal state. Moreover, *Process* schedules with $ta=0$ immediately its next internal event that has to be executed at simulation time $t=5$.

The control structure has been developed according to the RCP-concept successively. During this process the components *Control* and *Process* have been starting points for the control logic development. The components *RTC* and *PInterface* have been implemented afterwards. During the design phase the behavior of the real robot has been simulated and visualized by a separate component using the interface system *PInterface*. A detailed discussion of the robot application and other PDEVS-RCP examples can be found in [9].

7 Summary

Our research shows that the integration of Parallel DEVS and Real-Time DEVS to PDEVS-RCP based on the Simulation-based control approach fulfils the requirements of the RCP-concept. The PDEVS-RCP specification allows a successive development of simulation models beginning in the design phase till the operation phase. In contrast to RT-DEVS models PDEVS-RCP models do not require any specific simulation algorithms and as a result they can be simulated with ordinary PDEVS simulator and coordinator implementations. In addition, this opens the opportunity to use special run-time optimized execution algorithms based on the model-flattening approach published in [6].

The introduced PDEVS-RCP approach has been prototypically implemented in the scientific computational environment Matlab and has been tested with several laboratory applications. Key issues of further research are the investigation of PDEVS-RCP in the field of cooperative robot controls and its combination with the meta-modeling approach for re-configurable controls published in [5].

8 References

[1] Abel, D.; Bollig, A.: *Rapid Control Prototyping, Methoden und Anwendungen*. Berlin, Heidelberg: Springer Verlag, 2007

[2] Cho, S.M.; Kim, T.G.: *Real time simulation framework for RT-DEVS models*. Transactions of the Society for Computer Simulation Int., San Diego/CA, USA, 18(2001)4, 203 - 215

[3] Kremp, M.; Pawletta, T.; Colquhoun, G.J.: *Simulation-Model-Based Process Control of Discontinuous Production Processes*. In: Advances in Manufacturing Technology – XX, 4th Int. Conf. on Manufacturing Research (ICMR06), Liverpool, UK, Sept. 05-07, 2006, 49-54++

[4] Maletzki, M.; Pawletta, T.; Pawletta, S.; Dünow, P.; Lampe, B.: *Simulations-modellbasiertes Rapid Prototyping von komplexen Robotersteuerungen*. atp-Automatisierungstechnische Praxis, Oldenbourg Verlag, München, 50(2008)8, 54 - 60

[5] Pawletta, T.; Pawletta, S.; Maletzki, G.: *Integrated Modeling, Simulation and Operation of High Flexible Discrete Event Controls*. In: Proc. of Mathematical Modelling MATHMOD' 09, Argesim Report No. 35, Ed. I. Troch and F. Breitenecker, Vienna, Austria, February 11-13, 2009, 13 pages

[6] Zeigler, B.P.; Prähofer, H.; Kim, T. G.: *Theorie of Modeling and Simulation Second Edition: Integrating Discrete Event and Continuous Complex Dynamic Systems*. 2. Aufl. San Diego, San Francisco, New York, Boston, London, Sydney: Academic Press 2000

[7] Chow, A.C.-H.: *ParallelDEVS: A parallel, hierarchical, modular modeling formalism and its distributed simulator*. Transactions of the Society for Computer Simulation International, San Diego/CA, USA, 13(1996)2, 55 - 67

[8] Zeigler, B.P.; Kim, J.: *Extending the DEVS-Scheme knowledge-based simulation environment for real time event-based control.* IEEE Trans. on Robotics and Automation, 9(3), 1993, 351-356

[9] Schwatinski, T.: *DEVS-based Control of flexible Production and Robot Systems*. Wismar, University Wismar, Research Group CEA, Master-Thesis, January 2010