Flexible Task Oriented Robot Controls Using the System Entity Structure and Model Base Approach

T. Schwatinski*, T. Pawletta, S. Pawletta

Wismar University of Applied Sciences, Dept. of Mechanical and Process Engineering,
Res. Group Computational Engineering and Automation (CEA),
PB 1210, 23952 Wismar, Germany; *tobias.schwatinski@hs-wismar.de*

Abstract. A new method used for developing Flexible Task-oriented robot Controls (FTC) using the System Entity Structure (SES) is introduced. Task-oriented robot controls are based on the composition of atomic tasks aimed at achieving a previously specified goal. Flexible task-oriented controls differ in that the composition of atomic tasks is not fixedly predefined but is composed during the operation of the control based on actual process states and with respect to constraints in the sequence of tasks. The System Entity Structure is an ontology, which can be used for hierarchically representing system compositions. For cooperating robots the paper shows how to generate and execute FTCs specified by an SES and an associated model base (MB).

Introduction

Flexible Task oriented robot Controls (FTC) consists of several atomic tasks that are composed with respect to any constraints of their sequence with the aim of achieving a specific goal. The concrete sequence of atomic tasks can be determined either on the basis of actual process states during control operation or based on predictive process simulations. Therefore, FTCs belong [1] to intelligent robot controls and are related according to their implementation, due to requirements and complexity, to "large-scale" development [2]. As a consequence implementing such robot controls has to follow a systematic development process.

This paper presents the FTC/SES method used for systematically developing Flexible Task-oriented robot Controls (FTC) based on the System Entity Structure (SES) and Model Base (MB) formalism. The SES is a basic element of the FTC/SES method. The SES was originally developed in the eighties by Zeigler [3] and has been enhanced continuously to data engineering [4] until today.

The SES is an ontology designed for the hierarchical representation of real or imagined systems and is mostly used for defining meta models in the field of simulation technique. In our research the SES is used for specifying flexible industrial robot controls including subordinated process components. Using the SES the overall control task is divided into subtasks that are composed of atomic tasks and other composed subtasks. The declarative and modular, hierarchical specification of a control, including its process components using a SES, enables systematic control development. It also supports its reusability, adaption and maintenance.

Additionally, the FTC/SES method is based on the Simulation-Based Control (SBC) approach [5] and supports the successive development of simulation models within a homogeneous computing environment, beginning from the design phase until the operation phase. Below, the basics of the SES and the SBC are introduced in brief. Next using an example, their combined usage for specifying robot controls is discussed. Subsequently, the automatic generation of executable controls is shown. Finally, the paper summarizes important aspects of a prototype implementation and some experiences are summarized.

1 System Entity Structure and Simulation-Based Control Approach

1.1 The System Entity Structure

The System Entity Structure (SES) is an ontology. The SES forms a tree, the nodes of which can be categorized [4] as four node types: (i) entity, (ii) aspect, (iii) multiple-aspect and (iv) specialization. The general sequence of nodes in an SES is shown in Figure 1 (a). Entity nodes represent elements of the real or imagined world. Aspect nodes are used for decomposing entity nodes into finer-grained structures.

SNE Simulation Notes Europe – Print ISSN 2305-9974 | Online ISSN 2306-0271
SNE 22(2), 2012, 107-114 | doi: 10.11128/sne.22.tn.10135

Multiple-aspect nodes define multiplicity of entity nodes and specialization nodes represent categories or families of characteristics of entities. In addition attributes and their domain of definition can be attached to any node. Figure 1 (b) shows an example of an SES for specifying several automotive types. Every entity car consists according to the aspect node car of the entities engine, wheels and chassis. The entity engine is either specialized to the entity diesel or the entity gasoline and an attribute engine capacity is inherit to both. Moreover, the attribute stroke_cycle has been attached to the entity gasoline, which can be set to the values two or four. The entity 'wheels' is followed by a multiple-aspect node with a multiplicity of 4. Hence, this node will break down into four entities of type winter tire.

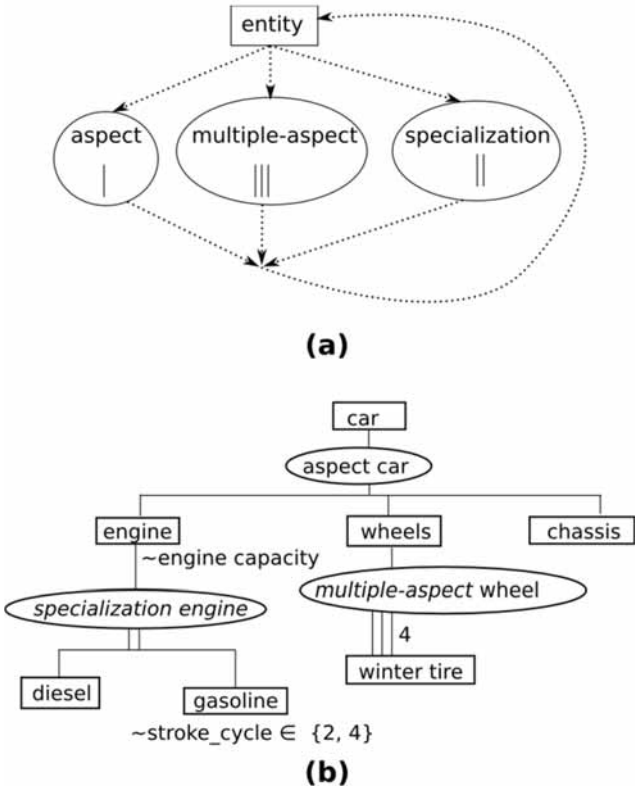

Figure 1. General structure of an SES.

Therefore the SES in figure 1 (b) characterizes the three different automotive types:

- diesel engine with a specific engine capacity, four winter wheels, chassis
- two-stroke cycle gasoline engine with a specific engine capacity, four winter wheels, chassis
- four-stroke cycle gasoline engine with a specific engine capacity, four winter wheels, chassis

In doing so, the SES can be used for clearly defining different characteristics of any composite system. All coupling relations between the entities have to be specified at aspect nodes. Moreover, the SES supports the specification of selection constraints. This means that the selection of an entity in a specialization may cause the selection of a certain entity in another specialization.

The simple example in Figure 1 does not define any constraints. Originally the SES was developed for specifying models in simulation technique field. The SES in combination with a model base (MB) that contains an executable software component for each leaf node of the SES allows simulation models to be generated automatically.

1.1 The Simulation-Based Control Approach

The Simulation-Based Control (SBC) approach described in [5] is a specific type of the software in the Loop (SiL) principle [6] and supports the usage of simulation models throughout during the entire development process of controls. Simulation models are enhanced stepwise beginning from the design phase to the automation phase and finally are used as control software directly during the operation phase using implicit code generation.

The control software (program) is executed using a real-time simulator. This approach means a development PC or industrial PC can be used to control real processes. The entire development process of controls based on the SBC approach is shown in Figure 2.

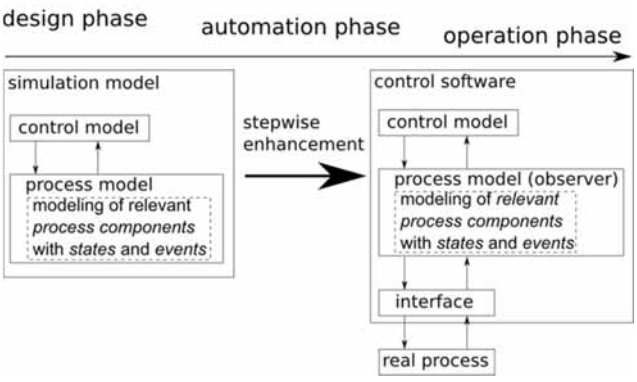

Figure 2. Simulation-Based Control (SBC) approach.

The SBC approach defines that any control software consists of a control model, an interface and, if required, a process model. The interface provides the connection between the real process and the control software.

This specific type of communication with a real process is called implicit code generation. The process model maps the behavior and states of real process components. In addition the process model that has already been used in the automation phase can be integrated into the control software as a state observer.

This procedure can increase the quality of controls, e.g. by calculating additional or immeasur-able state values. The control model maps the complete control logic.

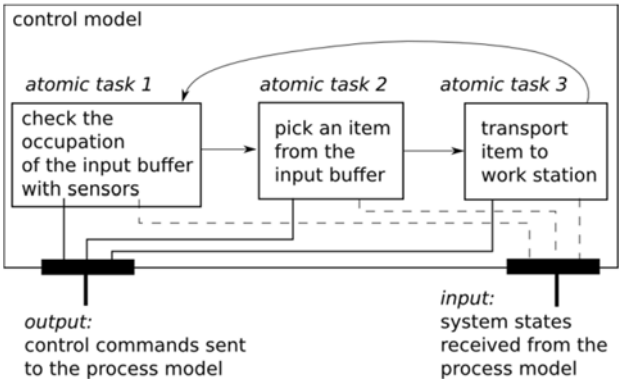

Figure 3. Coupling of atomic tasks in a control model following the SBC approach.

2 Developing Robot Controls based on a Declarative Specification

2.1 Integrating the SES/MB Formalism and the SBC Approach

The SBC approach supports effective implementation of robot controls from the beginning of the design phase to the end of the operation phase using simulation models. The SES/MB formalism supports a systematic and declarative specification of dynamic systems by means of a tree structure (SES) and automatic program generation using predefined param-eterizable modules from a model base (MB). In the following both approaches are used for defining task-oriented robot controls and it will be shown how highly flexible controls can be implemented.

The SBC approach supports following [5] the definition of task oriented controls. Predefined atomic tasks are composed and parameterized within a control model according to a specific control objective. In doing so any control commands and any reactions on system states are programmed in atomic tasks. This basic principle is shown schematically based on a simple control model in Figure 3. It shows that the sequence of atomic tasks is fixed within a control model. The whole flexibility of a control has to be implemented inside the atomic tasks and by means of their coupling relations. In particular complex and flexible robot applications comprise multifaceted relations between the atomic tasks within a control model as well as between the control model and the process model as a whole. This leads to highly complex controls because the SBC approach supports a structure-oriented modeling but unfortunately supports only a fixed composition of tasks.

The declarative specification of controls and the process-dependent generation of executable controls using the SES/MB formalism counteracts this drawback of the SBC approach.

Integrating the SES/MB formalism and the SBC approach for implementing Flexible Task-based Robot Controls is called the FTC/SES method. The major ideas behind this method are specifying a flexible control strategy with an SES and successively generating temporary controls during process operation. The most important elements and interactions of a flexible task-oriented robot control following the FTC/SES method are pictured in Figure 4. It is based on an adaptive control approach, which consists of a monitoring, a decision and a control generation and execution component. The monitoring component monitors every change in system states and the occurrence of control events during the execution of the temporary current control and continuously passes significant information to the decision component.

Based on the information received by a decision maker, the decision component checks whether the temporary control has to be adapted. If adaption is necessary, a new control structure is derived by analyzing the SES and afterwards passed to the control generation and execution component. The control generation and execution component generates and executes temporary control variants. A (model) generator creates a new temporary control. It evaluates the received control structure and builds the new control program using the predefined components in the model base.

According to the SBC approach, the control program consists of a control model C, a process model P and a process interface I and is executed by means of an event-oriented simulator that acts in real-time. The generated control program may be of limited lifetime or may be interrupted in consequence of real process events.

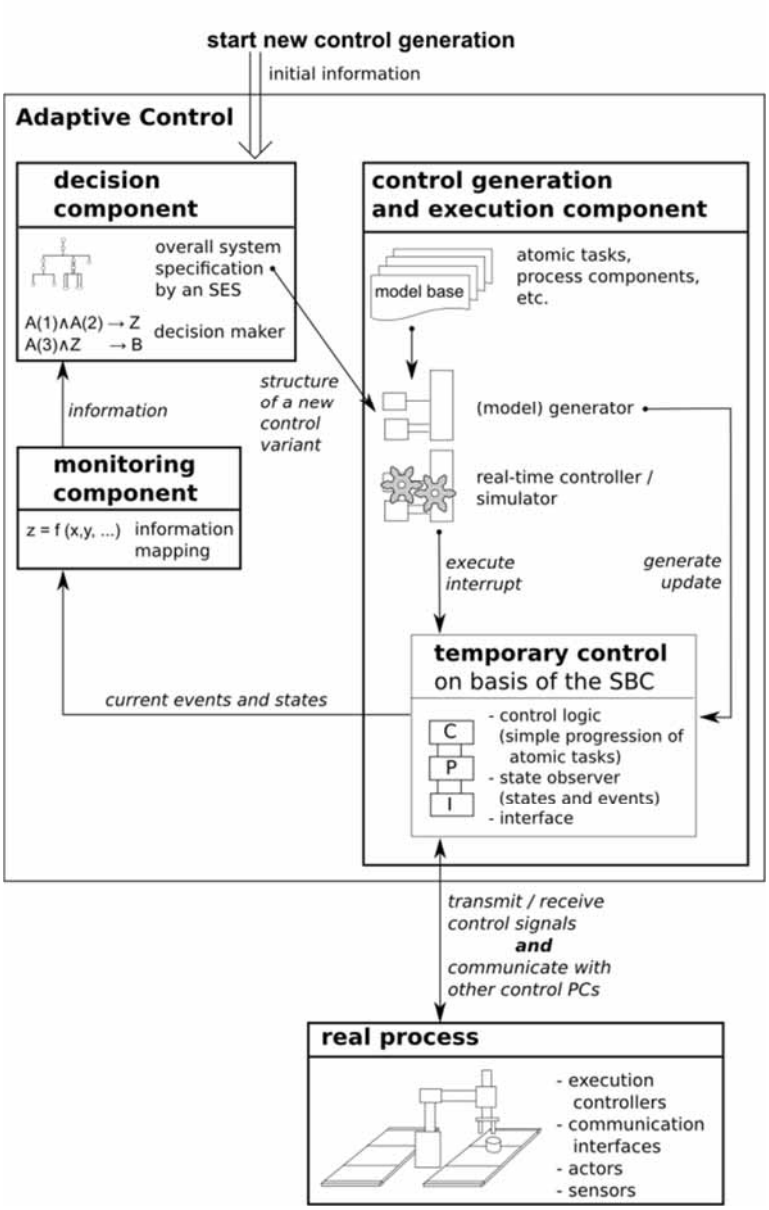

Figure 4. Elements and interactions of a flexible task oriented robot control following the FTC/SES method.

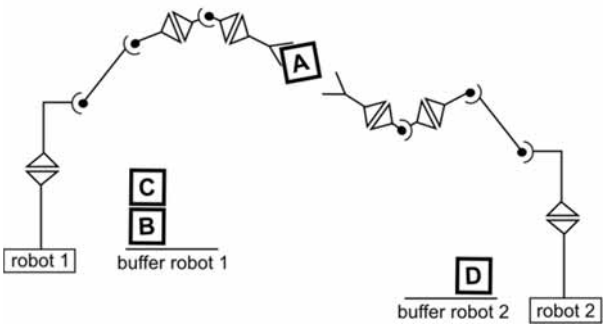

Figure 5. Cooperating robot application with two robots and two separated buffers.

In these cases a new control program is generated according to the described steps. This procedure is repeated until any predefined abort criteria occur. The declarative specification of robot controls and automatic generation of temporary control variants are discussed in detail in the following subsections.

2.2 Declarative Specification of Robot Controls

In this research we propose a slightly modified SES formalism called control-SES for specifying flexible industrial robot controls. The fundamental ideas are discussed using a small application shown in Figure 5. The application consists of two cooperating robots, each of which has a separate buffer.

The objective target for both robots is to re-arrange the objects in both buffers according to a user defined order. To fulfill this objective the robots have to cooperate, because objects are stacked in the buffers. The total amount of places in the storage areas is much smaller than the total amount of objects. Each robot has to cache objects from the other robot in its own buffer so that the other one can operate its necessary sort sequence. The transfer of objects takes place directly between both robots. At the beginning the control has to determine an optimal sort sequence to minimize the total amount of steps.

A simplified control-SES of the described robot application is shown in Figure 6. The pictured control-SES consists of two parts.

The upper part specifies the time invariant part of the control that according to the SBC approach defines a control model, a process model and a process interface. The overall task of the robot control is structured in several smaller control tasks and process interactions which are specified in the lower part. This part of the control-SES presents the time variant characteristics in terms of specialization nodes, which are used to specify the alternative use of atomic tasks in the control model and interactions between robots and buffers in the process model.

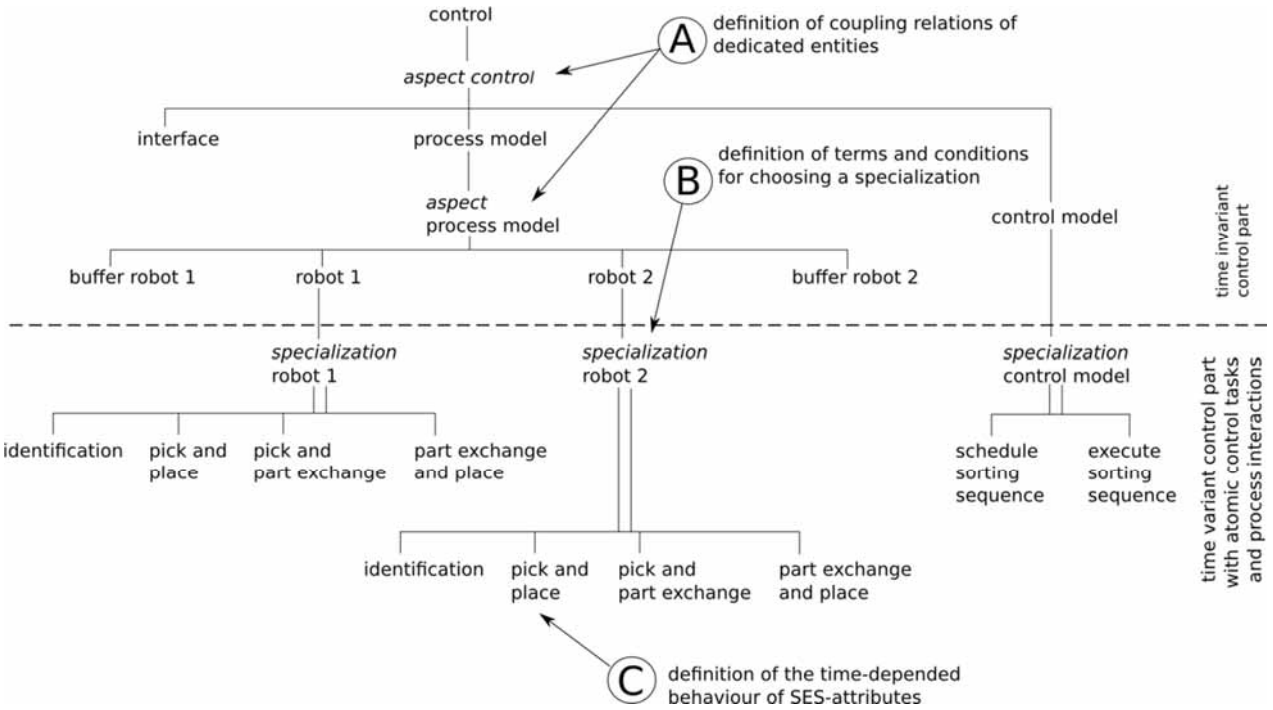

Figure 6. Declarative specification of a cooperative robot control using a control-SES.

The leaf nodes (C) of the control-SES are no further decomposable entities which are implemented as executable software components and stored in a model base (MB). The control-SES in Figure 6 is incomplete to preserve clarity. Beside nodes and edges a control-SES specifies node attributes. The leaf nodes representing atomic control tasks and process interactions define the modification of these attributes depending on the real process behavior. These attributes are described in more detail in the next subsection. The aspect nodes (A) define the coupling relations of the succeeding entities. Furthermore, the specialization nodes (B) define selection rules used for choosing dedicated atomic tasks and process interactions.

The general structure of all possible control variants follows from the time-invariant, upper part of the control-SES. A valid control variant is synthesized from the control-SES using a parameter vector that maps the current process behavior in terms of states and events to attributes of the control-SES. The result of this synthesis is a reduced tree structure in which all decision nodes like specialization or multiple-aspect nodes are resolved. Following [4] this procedure is called “pruning”.

During the pruning process all entity nodes in the undermost layer of the time invariant part of the control-SES are substituted with leaf nodes of the time variant part. Atomic control tasks and process interactions are selected in the specialization nodes. The structure of two temporary control variants synthesized from the control-SES pictured in Figure 6 is shown in Figure 7. The control structures pictured in Figure 7 represent only a subset of all valid temporary control variants according to the described application.

2.3 Sequence of Atomic Tasks and Process Interactions

The sequence of atomic control tasks and process interactions is determined during control operation on the basis of current process states and events. The flexibility of the control follows from the iterative synthesis and generation of temporary valid control variants. Therefore, the decomposition of the entire control in appropriate atomic control tasks and process interactions is a prerequisite. Specifying process interactions and defining their sequence is shown by robot 1. For this purpose, Figure 8 shows the description of entity node *robot 1* and its succeeding nodes in more detail.

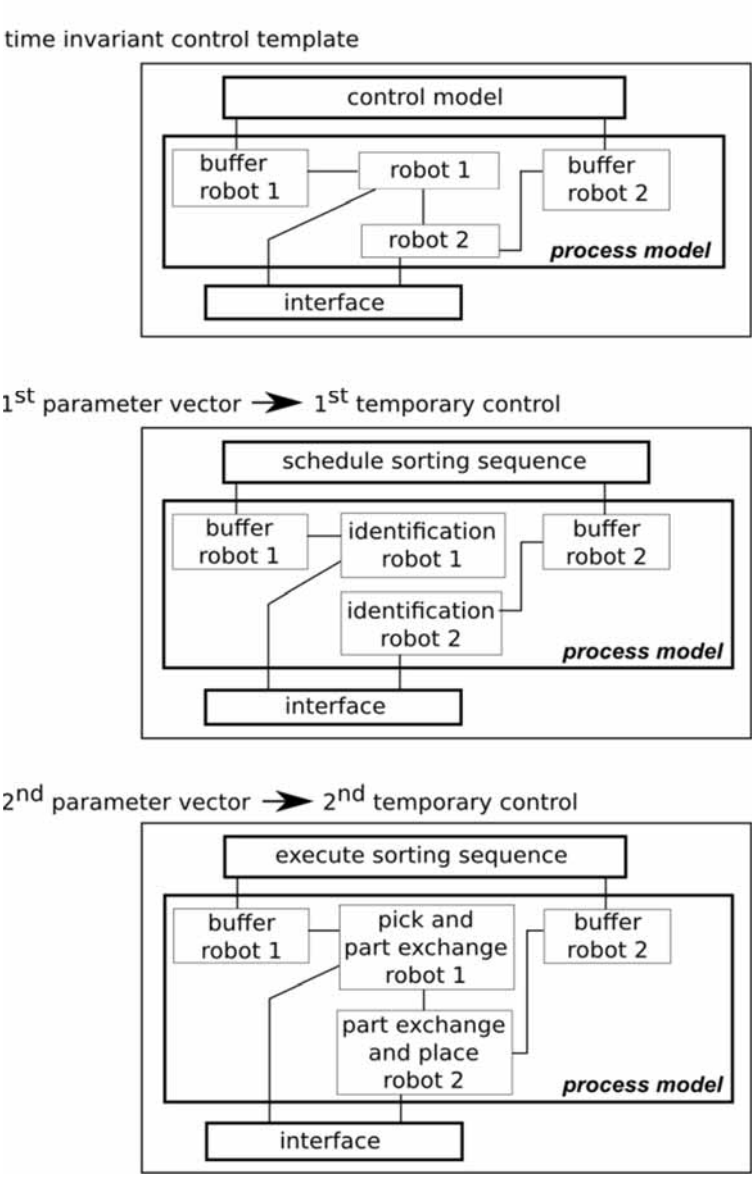

Figure 7. Generation of temporary control variants from a control-SES.

Inheritable parameters, including their domain of definition, are defined with attributes at entity node *robot 1*. In this example *roadmap* is a robot-specific set of points and connections in the configuration space of the robot used for path planning. The succeeding node *specialization robot 1* defines selection rules for process interactions evaluated in the control synthesis phase.

Every atomic control task or process interaction can define a time, state and event dependable behavior of SES-attributes using the operator "?".

If, for example, the interaction *identification* in Figure 8 is part of the current control, the SES-attribute *interactionRobot1* is declared changeable during control operation by the expression "interactionRobot1 = ?". In addition, the PRUNE action defines a new control synthesis if the value of *interactionRobot1* is set to one element of the set {*pickPlace, pickExchange, exchangePlace*}.

If, for example, a *pick and part exchange* interaction should follow after an *identification* interaction, then the SES-attribute *interactionRobot1* is set to *pickExchange* and a PRUNE action has to be performed.

That means a new control structure has to be derived by re-analyzing the control-SES using the modified SES-attribute *interactionRobot1*. In this case the specialized process interaction *pick and part exchange* is selected for entity node *robot 1* correctly.

3 Automatic Generation of Control Programs

Figure 9 shows the automatic generation of control programs. The starting point is an initial parameter vector P_{init} that contains configuration parameters and relevant process values that are partly mapped to SES attributes. At first the decision component analyzes the control-SES using the parameter vector P and derives a control structure in terms of a parameterized tree, called Pruned Entity Structure (PES).

The derived PES defines a unique control structure, where the leaf nodes present atomic control tasks and process interactions that are stored as parameterizable software components in a model base (MB).

Second the control generation and execution component generates an executable control program considering the information coded in the PES and using components from the MB.

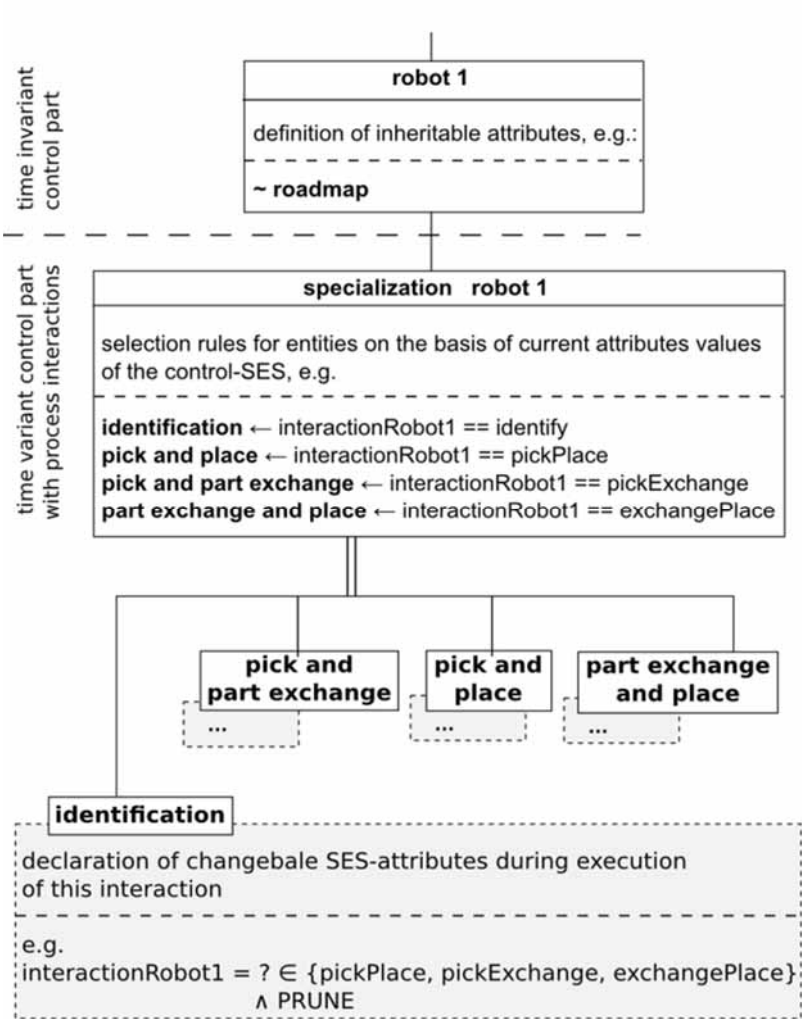

Figure 8. Extended specification of the control-SES for entity node robot 1.

This control program is structured according to the SBC approach in a control model C, a process model P and a process interface I, and is executed by means of a real-time simulator.

During control operation the atomic control tasks and process interactions modify state S of the temporary control according to the real process behavior. A subset of these state changes is monitored by the monitoring component that updates the parameter vector P too.

The decision component maps a subset of P to SES attributes A and decides whether the current temporary control is finished or has to be interrupted. If, so a new control structure is derived by analyzing the SES.

The cycle has to be repeated until the occurrence of any abort criteria. Moreover, Figure 9 illustrates the modularization used by the FTC/SES method. The strict separation of control specification on the one hand and implementation of atomic control tasks and process interactions as parameterizable components on the other supports the development of high flexible controls.

4 Summary

The FTC/SES method introduced has been prototypically implemented and tested in the programming environment MATLAB. The iterative pruning of the control-SES to derive valid control programs is implemented by a MATLAB interface to SWI-Prolog.

The atomic control tasks and process interactions stored in a model base have been implemented in MATLAB based on the DEVS formalism [7, 8]. Hence, each generated control program presents a modular hierarchical DEVS model that is executed by a DEVS simulation environment. This DEVS simulation environment has also been implemented in MATLAB and can be synchronized with real-time.

The application of a cooperating robot control as discussed has been implemented entirely using the introduced FTC/SES method. The interface component of the robot application has been implemented using the MatlabKK-Robotic Toolbox [9].

Finally, we conclude that the FTC/SES method simplifies the service introduction of flexible robot applications because any maintenance of control tasks and process interactions is focused only on strict encapsulated components in the model base.

Next, work will focus on developing further applications using the FTC/SES method to prove the approach.

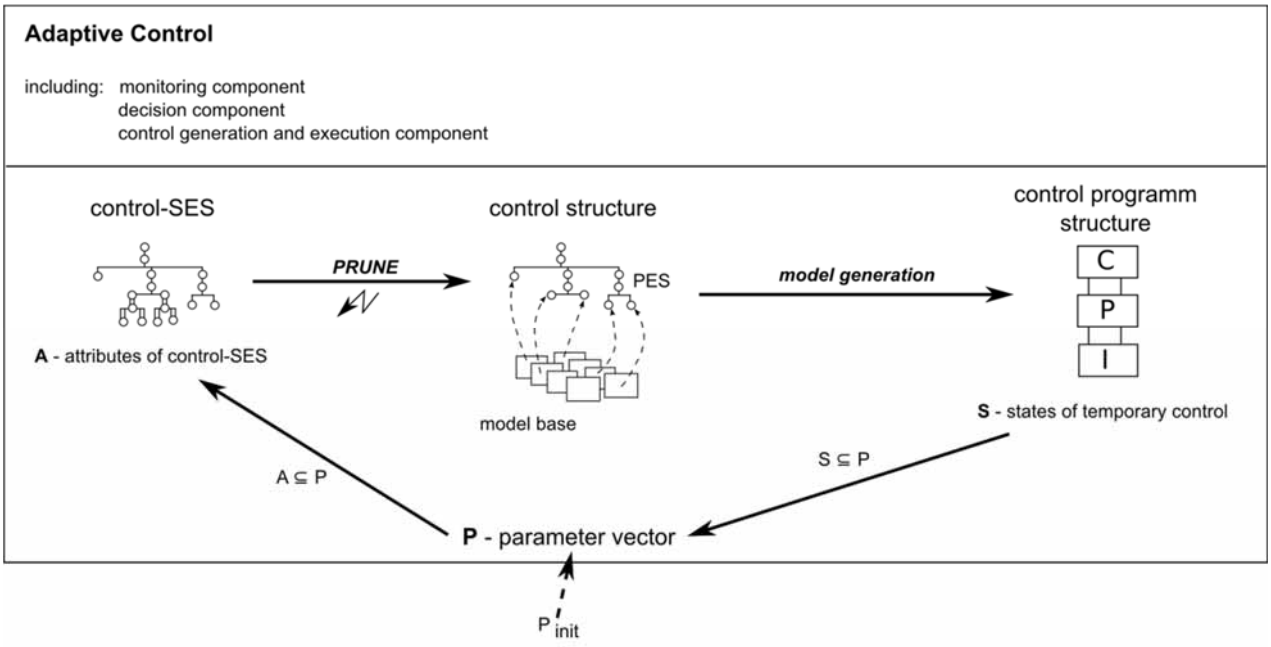

Figure 9. Automatic generation of temporary control programs.

References

[1] F. Breitenecker, D. Solar. *Models, Methods, Experiments - Modern aspects of simulation languages*. In: Proc. 2nd European Simulation Conference, Antwerpen, 1986, SCS, San Diego, 1986, 195 - 199.

[2] W. Jacak. *Intelligent robotic systems: design, planning, and control.* New York: Kluwer Academic / Plenum Publishers, 1998.

[3] M. Haun. *Handbuch Robotik.* Berlin, Heidelberg: Springer Verlag, 2007.

[4] B. P. Zeigler. *Multifaceted Modeling and Discrete Event Simulation.* Academic Press, 1984.

[5] B. P. Zeigler, P. E. Hammonds. *Zodeling and Simulation-based Data Engineering.* Burlington, San Diego, London: Elsevier Academic Press, 2007.

[6] M. Maletzki, T. Pawletta, S. Pawletta, P. Dünow, B. Lampe. *Simulationsmodellbasiertes Rapid Prototyping von komplexen Robotersteuerungen.* atp-Automatisierungstechnische Praxis, Oldenbourg Verlag, München.

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

[8] B. P. Zeigler, H. Prähofer, T. G. Kim. *Theory of Modeling and Simulation.*2nd Edition, San Diego, San Francisco, New York, Boston, London, Academic Press 2000.

[9] T. Schwatinski, T. Pawletta, S. Pawletta, C. Kaiser. *Simulation-based development and operation of controls on the basis of the DEVS formalism.* Proceedings of The 7th EUROSIM 2010 Congress, Prag, Czech Republic.

[10] M. Christern, A. Schmidt, T. Schwatinski, T. Pawletta. *KUKA-KAWASAKI-Robotic Toolbox for Matlab.* Hochschule Wismar, http://www.mb.hs-wismar.de/cea/sw_projects.html, 2011.

Submitted: September 2011
Revised: June, 2012
Accepted: August 1, 2012