Bridging the Gap between SysML and Design Space Exploration

Sivakumar Ganesan, Mauro Prevostini
ALaRI
University of Lugano
via G. Buffi 13, CH-6904 Lugano
Switzerland

Abstract

In the last few years the embedded systems design discipline required new design methodologies and new specification languages to support system engineers in developing heterogeneous systems where hardware and software are combined. One of the emerging modeling languages for system designers is the UML-based language called Systems Modeling Language (SysML). One of the most important tasks to be addressed early in the system design phase is the Design Space Exploration (DSE). DSE helps designers in discovering the optimal solutions among all possible combinations after mapping functional to architectural specifications. This paper describes an approach on how to use SysML for a DSE analysis within a system design phase.

1 Introduction

Advanced applications in the emerging areas like automotive systems or ambient intelligence require more and more sophisticated design methodologies and description languages. One of them is an emerging modeling language for system modeling called Systems Modeling Language (SysML) [Sys]. SysML is an extension of UML2.0 [Uml] to allow the modeling from a system engineering point of view and it is a joint initiative of the Object Management Group (OMG) [Omg] and the International Council on Systems Engineering (INCOSE) [Inc]. The SysML diagram taxonomy, which is described in details in [OMG06], reuses, extends and adds to UML diagram types as follows:

- UML diagrams that are reused, but are not extended: Use Case diagram, Sequence diagram, and State Machine diagram.
- UML diagrams that are reused and extended: Activity diagram (extends UML Activity diagram), Block Definition diagram (extends UML Class diagram), Internal Block diagram (extends UML Composite Structure diagram), and Package diagram (extends UML Package diagram).
- New diagram types: Parametric Constraint diagram, Allocation diagram/Allocation Traceability and Requirements diagram.

Thus SysML can be used in a very profitable way also for embedded systems design especially in early design phase where designers usually don't take into account hardware/software partitioning [VD05].
After the early design phase and once the system has been designed, the next step would be to map functionality of the system to architectural components. Once the mapping is done, a Design Space Exploration (DSE) can be performed to discover the optimal combinations between functional and architectural specifications, in relation with parameters like power consumption, performance, cost and size.
The goal of this paper is to show an approach which combines embedded systems modeling by means of SysML and DSE to help designers in evaluating the hardware/software partitioning solutions. In particular it shows how to use SysML in order to include information useful for the DSE phase.
The paper is organized as follows: Section 2 describes the state-of-the-art and related works in the area of UML combined with the DSE activity. Section 3 describes our approach on how to add information, useful for a DSE activity, into SysML diagrams. It also shows how to model systems for DSE by means of a mathematical structure. Section 4 illustrates our approach with a case study. In section 5 we conclude the paper and we briefly talk about future works. The last section is dedicated to the references.

2 State-of-the-art and related works

In this section we would like to describe the sate-of-the-art and the related works that combine UML with DSE. Herewith we show some of the approaches for performance analysis and DSE done with UML.
A design methodology that uses a UML Platform profile to model both application and platform was proposed in [CSM+02]. In this methodology, the analysis and synthesis tools available in the Metropolis framework [Lee04] were used to support a DSE. This work relies on simulation to obtain performance estimations. In [PW03] an approach for performance analysis using the UML-SPT profile [Gro02] is presented. In this approach, desired values for performance measures, such as response time, are added to the model. From this information, a performance model is derived, from which the performance analysis is computed, either by simulation or by analytical techniques. In this approach, all performance parameters should be annotated in the UML diagrams. The performance model answers questions about performance, like relationships between performance parameters. Natale et al. [NS03] also use the UML profile for real-time software. However, they address the schedulability analysis, which is fundamen-

tal for checking the correctness of hard real-time applications. This work discusses the use of fixed and dynamic priority scheduling mechanisms in designs that are developed using the UML-SPT profile.
The Real-Time UML profile [Sel00] defines a unified framework to express the time, scheduling and performance aspects of a system. It is based on a set of notations that can be used by designers to build models of real-time systems annotated with relevant QoS parameters. The profile standardizes an extended UML notation to support the interoperability of modeling and analysis tools but touches little on platform representation.
UML-RT [SR98] is a profile that extends UML with stereotyped active objects, called capsules, to represent system components. The internal behavior of a capsule is defined using statecharts; its interaction with other capsules takes place by means of protocols that define the sequence of signals exchanged through stereotyped objects called ports. The UML-RT profile defines a model with precise execution semantics, hence it is suitable to capture system behavior and support simulation or synthesis tools (e.g. Rose RT).
HASoC [GE02] is a design methodology based on UML-RT notation. The design flow begins with a description of the system functionality initially given in use case diagrams and then in a UML-RT version properly extended to include annotations with mapping information.
In [EDG+05] the authors discuss on annotations for quantitive analysis techniques used for the verification and validation of temporal characteristics of real-time embedded systems.
While in [GMTB04] the focus is on the concurrency issue and the authors describe how in the context of the UML parallelism should be modeled in a high level of abstraction.
In [PGL03] the discussion is on scheduling validation for UML-modeled real-time systems.

3 Our Approach

This section describes our approach which fits within a system engineering process. The work-flow used in our approach is described in Figure 1 and it starts with a requirements specification phase which includes a requirements analysis producing the refined requirements. The refined requirements are used as input for the system-level design phase where the designer has to describe the system by means of SysML diagrams like Internal Block, Requirements, Activity and Allocation diagrams. Details about how to use the SysML diagrams for a DSE phase will be discussed later in this section. Once the system design phase has been completed and all diagrams are available, the workflow continues with the generation of an XMI (XML Metadata Interchange) file from where the designer will extract the relevant data for the DSE phase. At this moment we need to build the design space by computing all the design points consisting of an *allocation* (selection of components), a *binding* (assignment of tasks to selected components) and a *scheduling* (execution order for the tasks). At this moment the designer has to compute the optimal Pareto-Points [BS80] that indicate which are the optimal solutions that are not comparable to each other, but the designer has to choose the most appropriate one according to his experience. Our effective contribution within this work-flow is indicated by the three dotted rectangles shown in Figure 1 called 'STEP 1', 'STEP 2' and 'STEP 3'. Because of the limited space, in this paper we will focus on 'STEP 1' which is based on [VD05] where the authors describe a possible usage of the SysML Assembly, Requirements, Activity and Allocation diagrams within a system design phase according to [SP05] which was the old SysML specification V0.9 DRAFT released in January 2005. In our paper we will show a similar usage of SysML, but according to [OMG06], namely the OMG final adopted SysML specification released in May 2006.
The idea of our approach is to propose how to represent DSE data within SysML diagrams, so that data can be extracted and reused by a DSE tool. Essentially we have to figure out how to represent *allocation* and *binding* by means of SysML diagrams, in particular using Requirement, Internal Block, Activity and Allocation diagrams. The *scheduling* could be represented by means of Sequence diagrams which we won't show in this paper due to a lack of space and because the related representation would be quite trivial.

3.1 Modeling Systems for DSE

This section deals with the formalization of the DSE with a six tuple structure M and mapping of the structure to SysML diagrams. The *DSE structure M* is defined as follows:

$$M = \{H, T, S, C, C_0, B\}$$

where:

- H is the set of available hardware components.
- T is the set of tasks that the system have to perform.
- $S \subseteq (H \times T)$ is a specification relation where for each hardware component $h \in H$ there is *at least* a task $t \in T$ such that $S(h,t)$. This relation describes the possible combinations of *allocation* and *binding*.
- C is the set of virtual components where $|C \setminus C_0| = |T|$. The virtual components are necessary in order to describe the allocation relation by means of SysML diagrams. See also point 5 later in this section.
- $C_0 \subseteq C$ is the set of communication busses used by $c \in C$ to exchange data.
- $B \subseteq (C \setminus C_0) \times T$ is a binding relation that must be total, that is, for every virtual component $c \in C$ there is a task $t \in T$ such that $B(c,t)$.

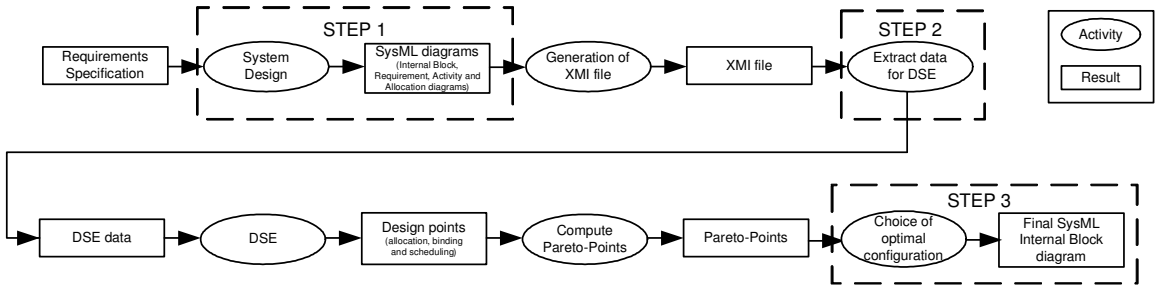

Figure 1: Work-flow used in our approach

The *DSE structure* M is then described by means of SysML diagrams. The steps to be performed are as follows:

1. Describe T by means of an Activity diagram.
2. Describe C and C_0 by means of an Internal Block diagram.
3. Describe H by means of a Requirement diagram. The Requirement diagram is used with the scope to describe the technical specifications of H by means of parameters like e.g. *Execution Time* and *Cost*.
4. Describe B by means of an Allocation diagram.
5. Describe S by means of an Internal Block diagram where each $h \in H$ is linked to a virtual component $c \in C$ according to the binding relation B.

The five steps described above are represented below in order to explicitly show how $M = \{H,\ T,\ S,\ C,\ C_0,\ B\}$ should be modeled. Assume that:

$$H = \{h_1, ..., h_k\}$$

is the set of hardware components available to the designer.

$$T = \{t_1, ..., t_n\}$$

is the set of tasks to be performed by the system.

$$S = \{(h_i,t_j)\}$$

is the specification relation $H \times T$.

Then

$$C = \{c_1, ..., c_{n+1}\}$$

is the set of virtual components where $C_0 = \{c_{n+1}\}$

$$B = \{(c_i,t_j)\}$$

is the binding relation where $(c_i,t_j) \in (C \setminus C_0) \times T$ such that $i{=}j$ with $i,j \in \{1, \ldots, n\}$.

The related SysML diagrams according to the above five steps will be expressed as explained in the following five sub-sections.

Activity diagram Figure 2 describes a possible task-chain by means of an Activity diagram that the system has to perform. The tasks are identified by $t_i \in T$ where $i \in \{1, \ldots, n\}$.

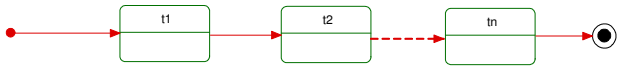

Figure 2: Activity diagram describing a possible task-chain

Internal Block diagram Figure 3 describes the virtual components by means of an Internal Block diagram. The virtual components are identified by $c_i \in C$ where $i \in \{1, \ldots, n{+}1\}$ and $c_{n+1} \in C_0 \subseteq C$. As you can

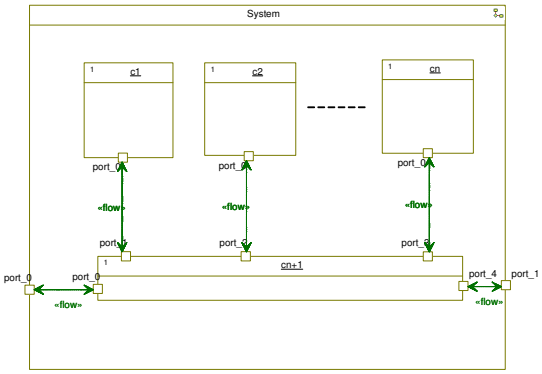

Figure 3: Internal Block diagram describing the virtual components of the System

see the virtual component c_{n+1} has the functionality to be the communication bus used by virtual components c_i where $i \in \{1, \ldots, n\}$.

Requirement diagram Figure 4 describes the available hardware components by means of a Requirement diagram. The requirements are identified by $h_i \in H$ where $i \in \{1, \ldots, n\}$.

Allocation diagram Figure 5 describes the binding relation by means of an Allocation diagram where each task $t_j \in T$ is allocated to a virtual component $c_i \in C$ such that $j = i$.

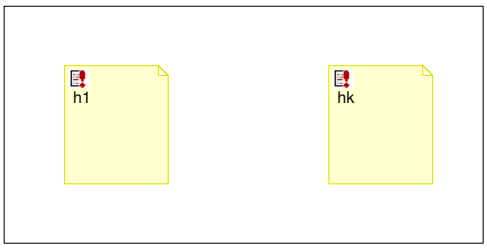

Figure 4: Requirement diagram describing the available hardware components

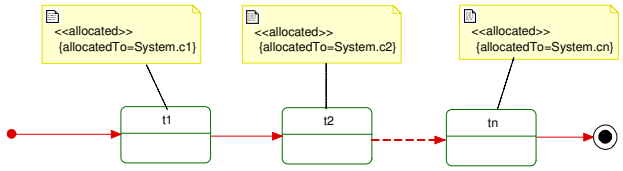

Figure 5: Allocation diagram describing the *binding* between tasks and virtual components

Final Internal Block diagram Figure 6 describes the specification relation by means of an Internal Block diagram where each hardware component $h_i \in H$ is binded to a virtual component $c_i \in C$ according to the Allocation diagram described in Figure 5 and the specification relation S.

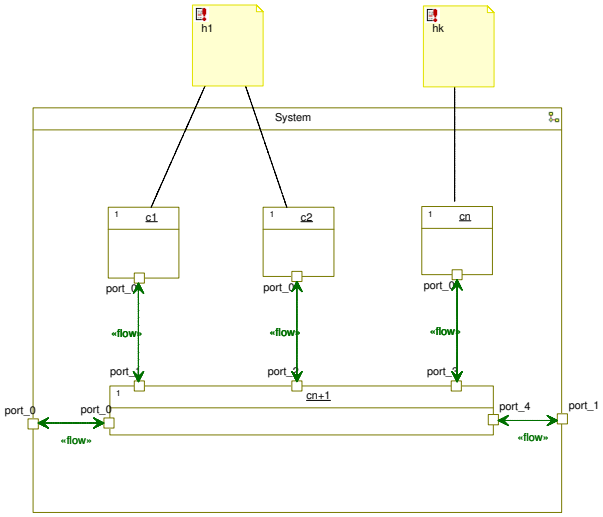

Figure 6: Final Internal Block diagram describing the *allocation* and *binding* between hardware- and virtual components

In Figure 6 we assume that

$S = \{(h_1,c_1), (h_1,c_2), \ldots , (h_k,c_n)\}$.

In this section we described a possible solution on how to represent *allocation* and *binding* by means of SysML diagrams. In order to represent *scheduling* we propose to use a set of Sequence diagrams with aim to express all the possible *allocation* and *binding* combinations. Because of a lack of space in this paper we won't show the Sequence diagrams.
In order to better explain our approach, we will show how to implement it in a simple case study explained in the following section.

4 Case Study

This section discusses a case study with the aim to describe a system by means of SysML diagrams following the formalism explained in the previous section.
Consider the task chain shown in Figure 7. It depicts a system specification with a set of tasks $t_i \in T$ that can be executed by different hardware components $h_i \in H$. The various components to execute the tasks together with the costs and the execution times are given in Figure 8. The only constraint that we impose on the system is that every hardware component executes the tasks sequentially. Also, at any given time a component can execute at most one task.

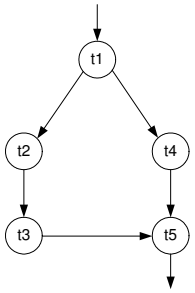

Figure 7: Task-chain of a system specification

Component	*Number*	*Cost*	*Execution time*				
			t1	t2	t3	t4	t5
h1: MIPS	1	150	3 *ms*	5 *ms*	--	--	4 *ms*
h2: DSP	1	100	--	--	12 *ms*	8 *ms*	6 *ms*
h3: FPGA	1	200	10 *ms*	--	7 *ms*	20 *ms*	--
h4: ASIC	1	350	--	--	--	2 *ms*	--

Figure 8: Available hardware components with costs and execution times for tasks

4.1 Implementation of our approach

Taking into account the constraints and the requirements of the system shown in Figures 7 and 8, we will now apply our approach by following the five steps described in Section 3.1 to perform DSE. To accomplish DSE, we start with the construction of the *DSE structure M*, which is given by

$$M = \{H, T, S, C, C_0, B\}$$

In the following subsections the possible values that each of the element in the *DSE structure M* can take is given. The following steps follow the 5 steps described in section 3.1.

Description of T The set of tasks is defined as follows:

$$T = \{t_1, t_2, t_3, t_4, t_5\}$$

and the related Activity diagram is shown in Figure 9.

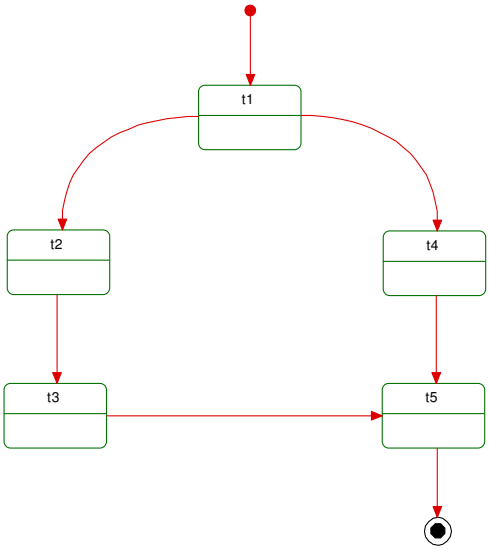

Figure 9: Activity diagram describing the task-chain

Description of C The set of virtual components is defined as follows:

$$C = \{c_1, c_2, c_3, c_4, c_5, c_6\}$$

where

$$C_0 = \{c_6\}$$

and the related Internal Block diagram is shown in Figure 10.

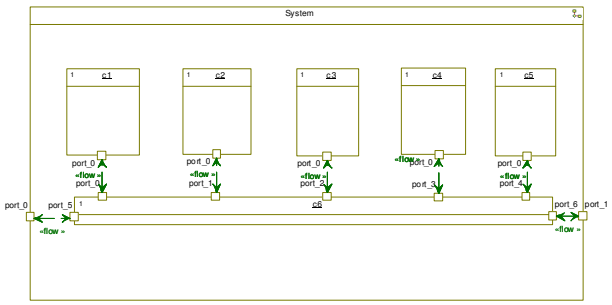

Figure 10: Internal Block diagram describing the virtual components

Description of H The set of hardware components is defined as follows:

$$H = \{h_1, h_2, h_3, h_4\}$$

and the related Requirement diagram is shown in Figure 11.

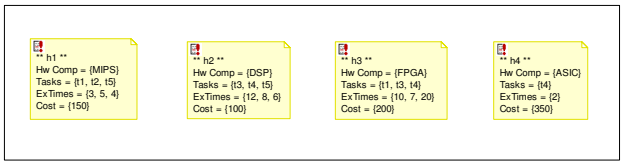

Figure 11: Requirement diagram describing the hardware specifications

Description of B The binding relation $B \subseteq (C \setminus C_0) \times T$ is defined as follows:

$$B = \{(c_1,t_1), (c_2,t_2), (c_3,t_3), (c_4,t_4), (c_5,t_5)\}$$

and the related Allocation diagram is shown in Figure 12.

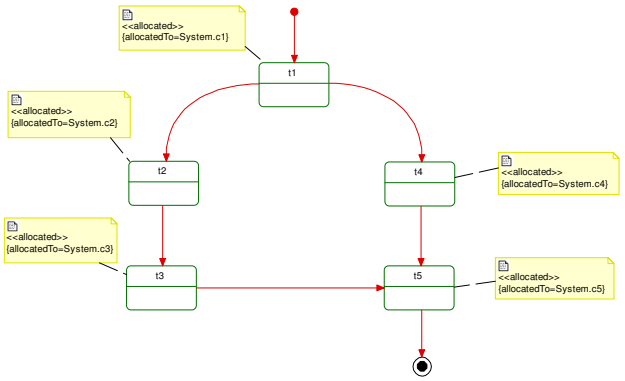

Figure 12: Allocation diagram describing the activities allocations

Description of S The requirement specification relation $S \subseteq (H \times T)$, according to Figure 8, is expressed as follows:

$$S = \{(h_1,t_1), (h_1,t_2), (h_1,t_5), (h_2,t_3), (h_2,t_4), (h_2,t_5), (h_3,t_1), (h_3,t_3), (h_3,t_4), (h_4,t_4)\}$$

and the related Internal Block diagram is shown in Figure 13.

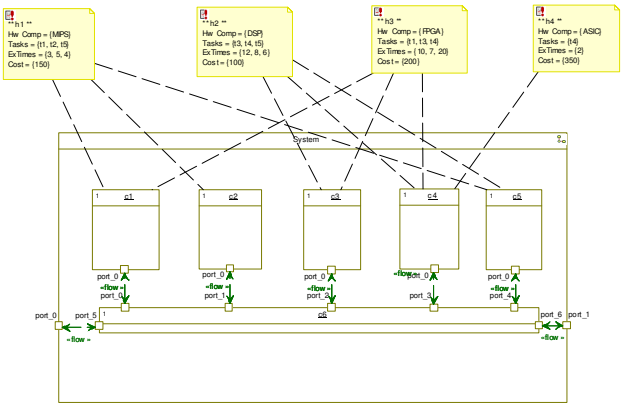

Figure 13: Final Internal Block diagram describing the specification relation

5 Conclusions and Future Works

In this paper we presented an approach to describe systems by means of SysML diagrams to address the DSE phase. The SysML diagrams are expressed according to the final adopted OMG specifications as described in [OMG06]. Our contribution is towards the system-level design using SysML to encourage designers of embedded systems to develop systems in an efficient way by performing DSE in the early stage of the design. Our approach aims to propose a way in order to have in one set of SysML diagrams all the possible combinations of the system configurations.
Since formal specifications provide a way to express problems without ambuiguity, we specified information relevant to DSE using a mathematical structure. This structure was then mapped on to the SysML diagrams. The main advantage of our approach is to have both DSE information and the system model in one repository.
In this paper we provided an approach to model DSE using SysML which corresponds to 'STEP 1' of Figure 1. As part of the future works we would like to automate the extraction of DSE information that means implementing 'STEP 2'. Having the design space modeled by means of a SysML Internal Block diagram as in Figure 13, the optimal configuration can be identified by extracting the information from the XMI file, which can be generated for example by the I-Logix UML/SysML tool [Ilo]. At this point 'STEP 2' could be performed and the designer would be able to compute the optimal Pareto-Points.
'STEP 3' would then be performed once the system architecture is chosen among the optimal solutions indicated by the computed Pareto Points. At that moment the designer would produce the corresponding SysML diagrams based on the Final Internal Block diagram shown in Figure 13.
In our work we used the UML/SysML tool named Rhapsody V6.1 from I-Logix [Ilo].

References

[BS80] R. Bayton and R. Spence. *Sensitivity and Optimization.* Elsevier, 1980.

[CSM+02] R. Chen, M. Sgroi, G. Martin, L. Lavagno, A. Sangiovanni-Vicentelli, and J. Rabaey. *Embedded System Design Using UML and Platforms.* In Proc. of Forum on specification and Design Languages, FDL'02, 2002.

[EDG+05] H. Espinoza, H. Dubois, S. Gérard, J. Medina, D.C. Petriu, and M. Woodside. *Annotating UML Models with Non-Functional Properties for Quantitative Analysis.* In Satellite Events at MoDELS 2005, 2005.

[GE02] P.N. Green and M.D. Edwards. *The modeling of Embedded Systems Using HASoC.* In Proc. of DATE 02, 2002.

[GMTB04] S. Gérard, C. Mraidha, F. Terrier, and B. Baudry. *A UML-based Concept for High Concurrency: the Real-Time Object.* IEEE - ISORC 2004, 2004.

[Gro02] Object Management Group. *UML Profile for Schedulability, Performance, and Time.* In OMG document n. ptc/02-03-02, 2002.

[Ilo] *http://www.ilogix.org.*

[Inc] *http://www.incose.org.*

[Lee04] E.A. Lee. *Advanced Tool Architectures.* Chess Conferences Reviews, UC Berkeley, 2004.

[NS03] M. Natale and M. Saksena. *Schedulability analysis with UML.* In UML for Real: Design of Embedded Real- Time Systems, Kluwer Academic Publishers, 2003.

[Omg] *http://www.omg.org.*

[OMG06] www.sysml.org OMG. *OMG Systems Modeling Language (OMG SysML) Specification.* Final Adopted Specification, 2006.

[PGL03] T.H. Phan, S. Gerard, and D. Lugato. *Scheduling Validation for UML-modeled Real-Time Systems.* ERCT 2003, 2003.

[PW03] D.C. Petriu and C.M. Woodside. *Performance analysis with UML: layered queueing models from the performance profile.* In UML for Real: Design of Embedded Real-Time Systems, Kluwer Academic Publishers, 2003.

[Sel00] B. Selic. *A Generic Framework for Modeling Resources with UML.* In IEEE Computer, 2000.

[SP05] www.sysml.org SysML Partners. *Systems Modeling Language (SysML) Specification.* version 0.9 DRAFT, 2005.

[SR98] B. Selic and J. Rumbaugh. *Using UML for Modeling Complex Real- Time Systems.* In White paper, Rational (Object Time), 1998.

[Sys] *http://www.sysml.org.*

[Uml] *http://www.uml.org.*

[VD05] Y. Vanderperren and W. Dehaene. *The SysML Profile for Embedded System Modeling.* In Proc. of Forum on specification and Design Languages, FDL'05, 2005.