Characterising Embedded Applications using a UML Profile

Sanna Määttä*, Leandro Soares Indrusiak†, Luciano Ost‡, Leandro Möller§, Manfred Glesner§,
Fernando Gehm Moraes‡, and Jari Nurmi*
*Department of Computer Systems
Tampere University of Technology, P.O.Box 553, FIN-33101 Tampere, Finland
Email: [sanna.maatta, jari.nurmi@tut.fi]
†Department of Computer Science
University of York, York, YO10 5DD, United Kingdom
Email: lsi@cs.york.ac.uk
‡Institute of Microelectronic Systems
Technische Universität Darmstadt, Karlstrasse 15, 64283 Darmstadt, Germany
Email: [moller, glesner]@mes.tu-darmstadt.de
§Catholic University of Rio Grande do Sul (PUCRS)
Av. Ipiranga, 6681 - P.32 - 90619-900, Porto Alegre, Brazil
Email: [ost, moraes]@inf.pucrs.br

***Abstract*—Application designers need to start the application design process before the final platform is available. Therefore, the designers need to have an abstract model of the platform at the early stages of the design process in order to validate the application functionality and evaluate its performance. Furthermore, platform designers need an application model to evaluate whether the computation and communication capacity of the platform is sufficient for the application. This paper identifies a minimalistic set of modelling constructs that can extensively characterise an application, which can be validated over a multicore Network-on-Chip (NoC) platform. The identified set of constructs is organized as a Unified Modeling Language (UML) profile in order to facilitate its use within UML-based design flows and tools. We present a practical application using the profile's constructs to model and constrain several subsystems of an autonomous vehicle control. Using the profile, we can cover sufficient aspects of the computation and communication requirements of the application, so that we can perform an extensive comparative analysis of alternative platform configurations very early in the design flow.**

I. Introduction

Design methodologies addressing today's complex embedded systems often start at too low level of abstraction. Starting the design at the abstraction level that is close to the implementation might result in a specification, which is burdened by unnecessary references to implementation. This can easily lead to under-optimised implementations. On the contrary, modelling and simulation at high abstraction levels yield early error detection and faster simulation due to less abstract models. Moreover, the separation of various aspects of the design, such as separating computation from communication and the application from the platform, allows us to explore alternative design solutions more efficiently [1].

Modelling and simulation are valuable methods for the beginning of embedded system design. Through modelling we get a better understanding of the system we are developing [2] and through simulation we can validate the system. Moreover, modelling reduces development time and costs, and supports technology development and optimisation [3].

Unified Modeling Language (UML) is a standard modelling and specification language [2] [4]. It is possible to extend UML in order to customise it for a particular domain by arranging the extension as a UML profile [5]. A profile is a set of stereotypes, constraints, and tagged values that are applied to specific model elements, such as classes, attributes, operations, or activities. When a profile does not modify the UML metamodel itself, the same modelling and design tools can be used and no new tools or extensions to the old ones are needed. The UML profile we present in this paper is based on a Profile for Modeling and Analysis of Real-time and Embedded Systems (MARTE) that customises UML for model-driven development of real time and embedded (RTE) systems and supports the specification, design, and validation stages of embedded system design [5] [6].

In this paper, we present the elements of an executable application model organised as a UML profile for embedded system design. The profile uses a subset of MARTE and in addition it constrains UML sequence diagrams in order to extensively characterise an application. The application model consists of any number of active and passive actors, whose communication order is defined by UML sequence diagrams. Furthermore, we show a case study of mapping the application onto a multicore Network-on-Chip (NoC) platform and validating the application together with the platform.

The rest of the paper is organised as follows: Section II describes related work and Section III presents further details about MARTE UML profile. Moreover, Section IV presents, which of MARTE's stereotypes we can use and

which additional elements our profile has. Next, Section V shows our case study of the system performance evaluation. Finally, Section VI concludes this paper.

II. Related Work

Many researchers are addressing application modelling using for instance a UML based workload model [7], a UML profile [8], task graph [9], or process network [10].

Kreku et al. model their application as a workload model using UML. This approach requires manual transformation of UML description into UML SystemC profile and futher into a new UML description, which includes only constructs available in SystemC [7]. Kukkala et al. have created a UML profile for embedded system design. Their profile covers the modelling of the application, platform, and mapping [8]. The profile is based on the UML profile for Schedulability, Performance and Time, which is supposed to be replaced by MARTE [5].

Jalabert et al. present an xPipesCompiler tool for instantiating application specific NoCs for heterogeneous multiprocessor systems-on-chip (MPSoC) [9]. In xPipes, applications are modelled as task graphs. Even if task graphs can show the dependency between tasks and denote their execution order, they abstract the temporal dimension and thus disregard important information about how often each task executes. Pimentel et al. present a framework for high level modelling, simulation, and performance evaluation of heterogeneous embedded systems [10]. They separate the application behaviour and architectural constraints at system level. Moreover, they use multiple Models of Computation (MoC), such as process networks for application modelling, dataflow graphs for model refinement, and discrete event for architecture simulation. For the application modelling they use Kahn Process Network (KPN) MoC. The application models can be created using a framework or manually from sequential C/C++ code.

III. MARTE UML Profile

MARTE's predecessor, UML profile for Schedulability, Performance, and Time (SPT) already extends UML with periodic tasks, schedulable objects, and timing and concurrency aspects [11]. However, MARTE is also capable of modelling processing elements, communication, task mapping, and real-time systems.

MARTE profile is organised into a set of packages, such as foundations, design model, and analysis model. MARTE foundation package contains basic elements for modelling non-functional properties, timing, and general resources. Moreover, an allocation concept associates application functions with the execution platform resources. The design model package includes elements for modelling generic components, software and hardware and components, and the application. Finally, the analysis package contains modelling capabilities for scheduling and performance analysis and enables designers to perform for instance timing analysis directly from the UML description instead of building a separate model for analysis [6].

TABLE I
STEREOTYPES USED IN OUR UML PROFILE

	Stereotype	**Used for**
MARTE	*<<SystemClock>>*	System simulation
	<<TimedEvent>>	Firing active actors
	<<RtUnit>>	Active actors
	<<PPUnit>>	Passive actors
EXTENSION	*<<Sequencing>>*	Sequence diagrams
	<<Message>>	Sequence diagram messages

IV. Application Modelling

The application modelling strategy we use in this paper follows the principles presented in [12]. An application is a set of communicating actors, whose communication is subject to explicit ordering of messages. Actors can be either active, which can initiate communication or passive, which may communicate only as a response to a received message.

We use a composite structure diagram and sequence diagrams in order to present the actors, their interconnection, and dynamic behaviour. Fig. 1 shows a composite structure diagram of the actors (big squares having a stereotype and the actors names inside them) and their input and output ports (small white and grey squares respectively). The lines connecting the input and output ports in Fig. 1 correspond the communication between the actors. All actors connected to a contiguous set of line segments participate on a communication pattern, and the ordering of the messages exchanged within each pattern is described by a sequence diagram.

Sequence diagrams consist of lifelines and messages. Lifelines represent the application actors and the messages between lifelines the communication between actors. The communication behaviour of the actors is constrained by using stereotypes and constraints.

As listed in Table I, we use MARTE's stereotypes *<<SystemClock>>*, *<<TimedEvent>>*, *<<RtUnit>>*, and *<<PPUnit>>* for defining the simulation period of the whole system, execution period of active actors, active actors, and passive actors respectively. In addition we need to create own stereotypes, *<<Sequencing>>* and *<<Message>>* for sequence diagrams and their messages.

Table II depicts the constraints we use for characterising our application model. We constrain our application actors and messages with period, data size, computation time, and priority, as can be seen in Fig. 1. Most of the constraints are used for workload modelling and the priority constraint for scheduling and arbitration. For the sake of clarity, in Fig. 1

TABLE II
CONSTRAINTS FOR THE UML SEQUENCE DIAGRAMS

Constraint	**Element**	**Used for**
Period	Active actor	Workload modelling
Time unit	Active actor	Workload modelling
Computation time	Message	Workload modelling
Time unit	Message	Workload modelling
Data size	Message	Workload modelling
Data size unit	Message	Workload modelling
Priority	Message	Scheduling and arbitration

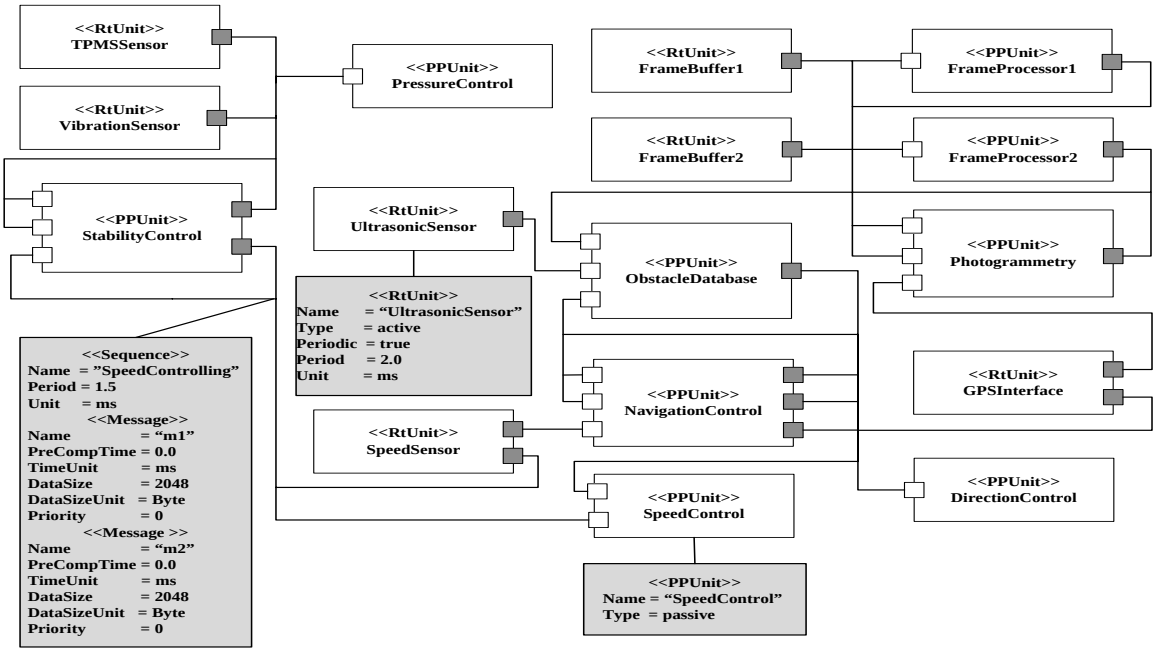

Fig. 1. Composite structure diagram of the actors

we show the constraints of only one active and passive actor, and one sequence diagram (see grey notes in the figure). The period constraint of the active actor defines how often it is executed, that is, how often it initiates communication. The data size constraint defines the packet size of the packet that carries the message in the NoC. The priority constraint can be used for scheduling and arbitration and the computation time sets the delay caused to the packet at the processing element as if a processor would execute the task.

Using the subset of MARTE and a few additional constraints organised as a UML profile, we can characterise an application in such a way that it is possible to evaluate whether a given multicore platform and a given mapping heuristic satisfy the computation and communication performance constraints of that application.

V. System Performance Evaluation Case Study

The example application we use is an autonomous vehicle, whose actors can be seen in Fig. 1. The vehicle is able to recognise unknown spaces by populating a database of obstacles obtained through stereo photogrammetry and ultrasonic sensors. The vehicle has two cameras, each of them feeding the system with 5 frames per second, each frame with a 352x288 resolution and 8 bit color representation. After the feature extraction on each of the images, photogrammetry estimates the distance from the vehicle to the obstacle and adds it to the database. Ultrasonic sensors also add information to the database, but since they have much shorter range, they are mainly used to refine the entries obtained with photogrammetry. The vehicle uses the obstacle database to guide its navigation process, which controls the speed and the direction of the vehicle. The stability control for the vehicle analyses its vibration, tyre pressure, and speed.

We created our application model using UML, which is not an executable language. Therefore, we use Ptolemy II framework for modelling and simulating our system. Ptolemy II is an actor-oriented framework for embedded system design and supports heterogeneous design with multiple Models of Computation [13]. Using Ptolemy II, we can assign execution semantics for both the application and platform models [14] in order to jointly validate them by simulating the system [12]. In our model, each application actor is represented by a lifeline in one or more sequence diagrams. The sequence diagrams are encapsulated inside Ptolemy II composite actors. For each message in a sequence diagram, the composite actor has input and output ports, to which the application actors are connected. Each application actor, that is, a lifeline is mapped onto one processing element in the platform. One processing element can have more than one lifeline mapped to it. Actors contain one or more tasks that are executed on the processing elements of the platform. For each task we have defined computational requirements by using the computation time constraint, which represent the workload the task imposes to the processor executing it. Our scheduling policy is nonpreemptive first in first served, which means that the processing element runs a task as early as possible to completion before the next task gets the processing capacity. This way we can handle multiple processes on a single processor with good accuracy.

We mapped the application model onto JOSELITO platform model [15] and simulated the system using 3x3, 3x4, and 4x4 mesh topologies and two different mappings. One of the mapping was totally random and the other one maps those actors that communicates the most with each other onto neighbouring processing elements. We used 50 MHz operation frequency and simulated the vehicle application for 18 seconds wall clock time. We measured the network latency for each packet, that is, how long a packet spends in the network either routed between network switches or in the input buffer of a

TABLE III
SIMULATION RESULTS IN MILLISECONDS

Topology		Random	Static
3x3	Critical communication	0,33	0,31
	All communication	2,13	2,10
	Critical tasks	2,22	0,36
	All tasks	3,17	1,98
3x4	Critical communication	0,31	0,31
	All communication	2,15	2,15
	Critical tasks	3,14	0,02
	All tasks	2,81	0,66
4x4	Critical communication	0,33	0,31
	All communication	2,20	2,16
	Critical tasks	1,92	0,02
	All tasks	1,67	0,44

switch waiting for routing. Furthermore, we also measured the time, which each task spends waiting for computation resources due to the nonpre-emptive scheduling policy.

On one hand this approach of considering both communication latency of the platform and computational requirements of the application allows us to explore the design space of the platform to make it meet the application requirements. On the other hand, application designer can explore, whether the target platform is sufficient for an application or its features and how much and what kind of resources are needed to run the application to meet its requirements.

Table III depicts the average latencies for critical communications and for all communications as well as average latencies for critical tasks and all tasks in milliseconds for both mappings (denoted as Random and Static in the table). We consider the parts of the vehicle that control its speed and direction as critical. The results show that the static mapping reduces the communication latencies a little, but has a significant effect on the time, which the tasks are waiting for the computation resources. This is mostly because the static mapping ensured that the actors involved in critical communication did not have to share the processing elements with other actors. The picture frames from the vehicle's cameras cause substantial communication workload, which can be noticed as longer average latencies of all communication than of critical communication.

Table IV shows the worst case latency of each message (denoted as M1-M18 in the table) of the sequence diagrams in milliseconds. Results depict that neither the mapping nor the topology have more than a minor effect on the worst case latency of each message and none of the configurations outpeforms the others.

VI. CONCLUSION

In this paper we presented a UML profile for characterising embedded applications. The profile is a subset of the MARTE UML profile, with some extensions that are necessary for extensive application describtion. We used the profile's constructs for modelling and constraining an autonomous vehicle application and validated the application on six different platform configurations. Based on the simulation feedback we are able to refine the system until we are satisfied with its performance. However, we should not let the platform performance restrict the application modelling and design. Instead, we should keep developing application models, which do not necessarily have a platform to run on and consequently set requirements for the future platform development.

REFERENCES

[1] K. Keutzer, S. Malik, R. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli, System Level Design: Orthogonalization of Concerns and Platform-Based Design, in *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 12, December 2000, pp. 1523-1543.

[2] G. Booch, J. Rumbaugh and I. Jacobson, *The Unified Modeling Language User Guide*, Addison Wesley, 1999.

[3] International Technology Roadmap for Semiconductors, Modeling and Simulation. 2007 Edition. Available at http://www.itrs.net/Links/2007ITRS/2007_Chapters/2007_Modeling.pdf (Referenced 03/2009).

[4] Object Management Group - UML, http://www.uml.org/ (Referenced 03/2009).

[5] UML Profile Specifications. Available at http://www.omg.org/technology/documents/profile_catalog.htm (Referenced 03/2009).

[6] Object Management Group, *A UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems*, Version Beta 2 2008. http://www.omgmarte.org/Documents/Specifications/08-06-09.pdf (Referenced 05/2009).

[7] J. Kreku, M. Hoppari, T. Kestilä, Y. Qu, J.P. Soininen, P. Andersson, and K. Tiensyrjä, Combining UML2 Application and SystemC Platform Modelling for Performance Evaluation of Real-Time Embedded Systems, in *EURASIP Journal of Embedded Systems*, vol. 2008, no. 3, 2008, pp. 1-18.

[8] P. Kukkala, J. Riihimäki, M. Hännikäinen, T.D. Hämäläinen, and K. Kronlöf, UML 2.0 Profile for Embedded System Design, in *Proceedings of the Design, Automation and Test in Europe*, 2005, pp. 710-715.

[9] A. Jalabert, S. Murali, L. Benini, and G. DeMicheli, xpipesCompiler: a Tool for Instantiating Application Specific Networks on Chip, in *Design, Automation and Test in Europe*, vol. 2, February 2004, pp. 884-889.

[10] A.D. Pimentel, C. Erbas, and S. Polstra, A Systematic Approach to Exploring Embedded System Architectures at Multiple Abstraction Levels, in *IEEE Transaction on Computers*, vol. 55, no. 2, February 2006, pp. 99-112.

[11] Object Management Group, *UML Profile For Schedulability, Performance, and Time*, Version 1.1 2008. http://www.omg.org/technology/documents/formal/schedulability.htm (Referenced 05/2009).

TABLE IV
WORST CASE LATENCY FOR EACH MESSAGE IN MILLISECONDS

Message	3x3		3x4		4x4	
	Random	Static	Random	Static	Random	Static
M1	0,03	0,03	0,03	0,03	0,03	0,03
M2	4,12	3,97	3,97	3,97	3,97	3,97
M3	3,97	3,97	3,97	3,97	4,46	3,97
M4	3,97	0,38	0,38	0,38	0,37	0,38
M5	0,26	0,38	0,38	0,38	0,38	0,38
M6	0,10	0,16	0,10	0,11	0,11	0,11
M7	0,01	0,01	0,01	0,01	0,03	0,01
M8	0,21	0,11	0,11	0,11	0,13	0,11
M9	0,03	0,03	0,03	0,03	0,05	0,03
M10	0,10	0,05	0,05	0,05	0,05	0,05
M11	0,42	0,42	0,42	0,42	0,45	0,42
M12	0,03	0,03	0,03	0,03	0,03	0,03
M13	0,01	0,01	0,03	0,01	0,01	0,01
M14	0,06	0,07	0,07	0,05	0,10	0,09
M15	0,05	0,05	0,05	0,05	0,05	0,05
M16	0,03	0,05	0,03	0,03	0,06	0,05
M17	0,03	0,03	0,03	0,03	0,03	0,03
M18	0,03	0,02	0,01	0,02	0,01	0,04

[12] S. Määttä, L.S. Indrusiak, L. Ost, L. Möller, J. Nurmi, M. Glesner, and F. Moraes, Validation of Executable Application Models Mapped onto Network-on-Chip Platforms, in *International Symposium on Industrial Embedded Systems*, France, June 2008, pp. 118-125.

[13] J. Eker, J.W. Janneck, E.A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neundorffer, S. Sachs and Y. Xiong, Taming Heterogeneity – the Ptolemy Approach, in *IEEE*, vol. 91, no. 1, 2003, pp. 127-144.

[14] L.S. Indrusiak, A. Thuy, and M. Glesner, Executable System-Level Specification Models Containing UML-based Behavioral Patterns, in *Design Automation and Test in Europe (DATE)*, 2007, pp. 301-306.

[15] L. Ost, L. Möller, L.S. Indrusiak, F. Moraes, S. Määttä, J. Nurmi, and M. Glesner, A Simplified Executable Model to Evaluate Latency and Throughput of Networks-on-Chip, in *Symposium on Integrated Circuits and Systems Design*, 2008, pp. 170-175.