# An Embedded System Modelling Methodology for Design Space Exploration

F. Herrera, P. Peñil and E. Villar

University of Cantabria

F. Ferrero and R. Valencia

GMV

**The design of embedded systems is being challenged by their growing complexity and tight performance requirements. A synthetic, but sufficiently detailed model of the system and its environment, fast performance assessment technologies, and smart exploration of design alternatives are required for a fast assessment of the optimum design alternative. This paper focuses on the former aspect, by presenting the COMPLEX UML/MARTE modelling methodology, which introduces novel and necessary aspects for speeding up architectural exploration. This modelling methodology has been integrated into the COMPLEX Design Exploration framework, which has served to show through a demonstrative example (an EFR vocoder) the applicability and benefits of the proposed modelling approach.**

**Keywords**

UML, MARTE, modelling, Design Space Exploration, ESL, System Level Design.

## I. INTRODUCTION

The design of embedded systems is in a highly competitive context. The translation of a successful design into a successful product highly depends on becoming the first product in the market with new complex functionalities fulfilling tight performance constraints. In this scenario, the task of system engineers becomes challenging. They require an early assessment of the design alternatives, since about 90% of the overall costs are determined at the first stages of the design [1]. At the same time, a right assessment becomes difficult due to the complexity of applications and platforms, whose performance is characterized by a diverse set of factors, such as the software architecture of the application; the architecture of the HW platform; how the application functionalities are executed by the processing resources of the platform; cache sizes; memory sizes; and many others. Design Space Exploration (DSE) is a key design activity [2] in charge of enabling such an early assessment. A DSE framework has three main requirements: (1) a modelling methodology suitable for Design Space Exploration (DSE), (2) tools able to produce fast and sufficiently accurate performance metrics, and finally (3) an exploration engine able to perform a smart search of the overall design space. This paper focuses and contributes to the first point.

A set of methodologies have enabled the capture of the system architecture and main system parameters. A special effort has been done to develop methodologies based on the Unified Modelling Language (UML) [3], supported by specialized profiles, such as SysML and MARTE [4]. The MARTE profile is an OMG standard that offers a rich set of extensions specifically suited for the specification of embedded real-time systems. MARTE enables building models containing detailed information about the platform attributes and the system architecture for enabling performance analysis. However, the proposed system specification methodologies are still limited for DSE purposes. In these methodologies, the exploration of different platform architectures, of different architectural mappings, and even a small change in a design parameter (e.g., a cache size) requires a manual change of the model. Moreover, when the model is used for producing an executable virtual system for a simulation-based performance analysis, a regeneration of the executable model is typically required. Model edition and regeneration of the executable performance model add to the simulation time at each iteration of the DSE loop, and they have a non-negligible impact in the exploration time, up to a point which can make the exploration of a sufficiently wide design space unaffordable. Current modelling methodologies also lack ways for capturing within the model the output performance metrics to be used by the objective function(s) of the DSE process. Enabling their capture in the model in a tool independent manner would enable the direct relation of such metrics with the performance constraints also captured in the model. Performance constraints are mandatory, thus they impose a frontier for the solutions to be considered as feasible solutions.

This paper presents a component based modelling methodology based on UML/MARTE and explicitly designed for supporting DSE. The methodology has been developed as the entry point of the COMPLEX framework [5], especially suited for DSE. Specifically, the methodology supports the specification of a *design space*, i.e., a set of design solutions, rather than a single design solution. Moreover, the methodology supports also explicit constraints and rules that limit the space of solutions to those that can be of interest. The methodology also supports the specification of a set of output performance metrics, which are required as inputs to the objective functions and/or to define associated constraints on them. The methodology is supported by a specific toolset developed in the COMPLEX project. This toolset enables the automatic generation, directly from the COMPLEX UML/MARTE model, of an executable and configurable performance model. This performance model is based on the SCoPE technology [6], which through a fast simulation, enables functional validation and provides a rich set of performance metrics. The SCoPE performance model is configurable and represents the design space captured at the model. In each DSE iteration, before launching the

simulation, the executable model is configured, by giving specific values to design parameters (e.g. a given cache size), and defining a specific platform architecture and architectural mapping. Each new configuration, which represents a new design point to be explored, requires neither an edition of the UML/MARTE model nor a regeneration of the executable performance model. The simulation of the performance model for each design point provides performance metrics to a DSE exploration tool (developed by COMPLEX partners as an evolution of the Multicube Explorer [7] tool) which decides the next design point to be explored. The explanation of the whole COMPLEX toolset is out of the scope of this paper. In this paper, we focus on the modelling methodology.

The structure of the paper is as follows. Section II will present the related work. Then section III introduces the proposed methodology, presenting its main concepts and showing how a system model is built. Section IV explains the features of the methodology for specific support of DSE, which constitutes the main contribution. Section V shows the benefits of the modelling approach through experimental results. Section VI gives the main conclusions of this work.

## II. Related Work

Despite the relative recent development of the MARTE profile, several works have proposed UML/MARTE based methodologies. Gaspard2 [8][9] is a design environment for data-intensive applications which enables a MARTE description of both, the application and the hardware platform, including MPSoC and regular structures. Gaspard2 uses composite diagrams and the MARTE profile for capturing both, application and platform architectures. Gaspard2 tooling supports the chaining of different model to model (M2M) transformation tools. This facilitates the generation of synthesis flows, and also of performance models. Specifically, Gaspard2 supports the generation of SystemC TLM models at the Programmers View Time (PVT) level. It enables fast simulations, which speeds up exploration. However, a change in a parameter or in the architecture requires the edition of the model and moreover, the re-generation of the TLM model.

MoPCoM [10] is another design methodology for the design of real-time embedded systems which supports UML and the MARTE profile for system modelling. Specifically, MoPCoM uses the NFP MARTE profile for the description of real-time properties; the HRM MARTE profile for platform description; and the Alloc MARTE profile for architectural mapping. Moreover, MoPCoM defines three levels of generation. From all of them, the second level, called Execution Modelling Level (EML), targets the generation of models for performance analysis, and it is suitable for obtaining performance figures used in DSE iterations. However, work reported in [10] mostly focuses on the Detailed Modelling Level DML level, intended for implementation, by enabling VHDL code generation. Like in Gaspard2, exploration of architectural alternatives requires the edition of the UML/MARTE model and a re-generation of the executable performance model.

The work of [11] proposed a UML/MARTE based methodology in order to reduce the effort to capture the set of architectural alternatives for design space exploration. For it, instead of relying on an element explicitly representing the allocation (e.g. an UML association with the MARTE <<allocate>> stereotype), [11] introduces *activity threads*. An activity thread (AT) is a UML activity diagram where each path reflects a design alternative, that is, an architectural mapping.

Co-Fluent methodology [12] provides a modelling methodology relying on SysML and on MARTE which uses the <<*assign*>> stereotype for expressing allocations. However, they are used for modelling a single allocation, thus a single implementation alternative. The methodology captures application and hardware architecture by means of composite diagrams and SysML blocks. UML activity diagrams are used to specify application execution flows. The MARTE HRM profile is used for capturing the HW platform.

In [13], a methodology for supporting designers on the evaluation of the HW/SW partitioning solutions, specifically, to identify design points fulfilling the timing constraints is shown. It proposes a way to depict in one set of diagrams all possible combinations of system configurations. By means of annotation of MARTE non-functional properties and of the application of schedulability analysis, the design space is restricted to the design points fulfilling timing requirements. However this methodology neither reports optimum solutions, nor it relies on automated technologies for the estimation of performance metrics.

## III. Modelling Methodology

### A. Introduction

The system modelling methodology described in this paper is follows a component-oriented approach [14] and applies the Model Driven Architecture (MDA) [15] principles in the development of HW/SW embedded systems. Moreover, the proposed approach makes this methodology *software centric* [16] as it enables the description of a platform independent model (PIM) which can be fully allocated to a SW implementation, and thus can be considered as an *application model*. However, the methodology also enables to consider the HW implementation of application components.

In Component-based Software Engineering (CBSE) [14], the system is built as a composition of application components interacting with each other only through well-defined interfaces. Components are software units that exhibit their interfaces (provided or required). This way, the application can be split into clearly separable and reusable blocks, improving the organization of the product as well as its reusability and modularity.

The COMPLEX methodology supports the separation of concerns. This separation is achieved by providing distinct system views to the designer, in the shape of UML packages, each one for every relevant aspect:

- **Data View:** captures the relevant data types of the system, i.e., types of data exchanged at interfaces.
- **Functional View:** captures the functional structure of the system, as a set of interfaces and a set of classes implementing and using those interfaces.
- **Communication and Concurrency (CC) View:** captures the application architecture, enclosing classes into components. It also captures the non-functional aspects of system functionality related to the application behaviour, such as concurrency and real-time constraints.
- **Platform Description View:** describes both software and hardware resources of the platform.
- **Architectural View:** describes the platform architecture and the architectural mapping of the application components onto platform processing resources. It is also the view where the DSE parameters, rules and constraints that will enable the exploration of the different architectural solutions are captured.
- **Verification View:** is devoted to the definition of the system stimuli environment. Stimuli modelling is explained in detail in [17].

### *B. Description of the Application (PIM)*

The initial steps in the methodology consist of identifying the system functions, modelled through the Data model and Functional View. System functions are captured by means of the use of UML use cases and the relations between them. The UML use cases will allow the designer to identify the UML interfaces that model the system functions and create the UML classes that would implement them. System functions are modelled by means of UML interfaces stereotyped with the MARTE *<<ClientServerSpecification>>* stereotype. The operations of the *<<ClientServerSpecification>>* interfaces might have parameters whose types should have been defined in the Data Model.

In the CC view, the application components are defined. The CC view contains omponents with the MARTE *<<RtUnit>>* and *<<PpUnit>>* stereotypes. The former stereotype identifies a component which an have its own execution thread, providing/requiring services to/from others components by means of its provided and required interfaces. The latter represents a non-active component, which provides services, such as giving access to shared data, as a reaction to active component demands. The functional behaviour of the UML components is defined by adding instances of functional classes. They are captured as UML properties of the UML component. Property type must be one of the classes defined in the Functional View. UML Components use UML Ports stereotyped with the MARTE *<<ClientServerPort>>* and *<<RtFeature>>* stereotypes for defining the interfaces which serve for communicating with other components, and for specifying real time behaviour. In order to complete all the necessary information for the CC View, the user will add an additional UML component with the COMPLEX *<<system>>* stereotype. This component represents the PIM and captures the software architecture. It includes application component instances, captured as UML properties typed as any of the <<RtUnit>> or <<PpUnit>> components declared in the CC view and UML connectors for their interconnection.

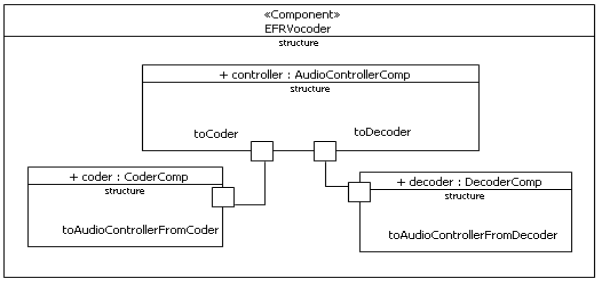


**Figure 1.** Arquitecture of the EFR vocoder Application.

The composite diagram of Figure 1 shows the SW architecture of an EFR vocoder system, within a <<system>> component of the CC view. It encloses three component instances ("coder", "controller" and "decoder"). Each instance (e.g. "coder") is an UML property typed as a component previously defined in the CC view (e.g. "CoderComp"). Such a component, in turn, encloses an instance of a functional class defined within the functional view (e.g., "Coder").

### *C. Platform and Architectural Mapping (PSM)*

The proposed modelling methodology supports the description of the HW/SW platform within two views, the Platform view and the Architectural view. SW and HW components are declared in the Platform view. An RTOS is declared as a component with the MARTE *<<Scheduler>>* stereotype. Declaration of HW components relies on the MARTE Hardware Resource Modelling (HRM) subprofile (i.e. *<<HwProcessor>>*, *<<HwBus>>*, *<<HwCache>>*, *<<HwRAM>>*,…).

The architecture of the platform is captured within the Architectural view, specifically within a UML component ("archi_system" in Figure 2), stereotyped with the COMPLEX <<system>> stereotype again. This component contains SW component instances and the hardware platform architecture, as reflected in Figure 2. Figure 2 also shows how a fixed architectural mapping can be specified. Application component instances (the three on top of Figure 2) captured the CC view are referenced in the Architectural view (for it, first, the *<<system>>* component of the Architectural view is captured as an extension of the <<system>> component of the CC view). Then, associations with the MARTE *<<Allocate>>* stereotype, which reflects a spatial allocation, map application components onto instances of platform components with computation capabilities.

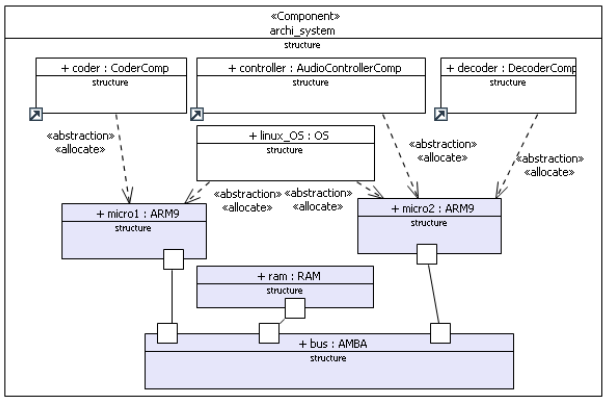


**Figure 2.** Architecture of the EFR Vocoder system.

## IV. MODELLING FEATURES FOR DESIGN EXPLORATION

The diagram of Figure 2 models a single implementation since it shows a fixed architecture, a single architectural mapping, and fixed attributes for platform components. The COMPLEX UML/MARTE modelling methodology provides features for enabling the specification of a design space, that is, a set of design alternatives. This enables the generation of a configurable executable counterpart for obtaining performance estimates. This way neither model modification, nor performance model regeneration is required. This makes the methodology suitable for DSE flows.

### A. Specification for Design Space Exploration

The COMPLEX UML/MARTE modelling methodology enables the specification of a design space which can consist of (a) a set of architectural mappings (allocation space); (b) a range of values for platform attributes (attribute values space); and (c) a set of platform architectures (architecture space). Moreover, the design space can be shaped and constrained through the definition of DSE constraints and rules. Finally, the methodology also enables the definition of the output metrics to be considered for the goal functions used in the exploration. A dedicated UML profile, the COMPLEX profile, has been created to add the necessary semantics that are missing in the MARTE profile with regard to the aforementioned features. Therefore, this UML profile complements MARTE for DSE. Four elements have been created to represent the necessary concepts: *Exploration parameters*, of three possible types (allocation parameters, scalar parameters and vector parameters); DSE *Rules*; *Constraints* and *Estimation parameters* (for defining output metrics).

### B. Definition of the Design Space

#### 1) Specification of a Space of Allocations

Architectural mapping is a factor with a big impact on performance. This methodology enables the description of a set of architectural mappings. This set is captured through one or more UML comments placed in the Architectural view, and stereotyped with the MARTE *<<Assign>>* stereotype (to specify the allocation itself) and with the COMPLEX *<<DseAllocationParameter>>* (which provides a name to the allocation).

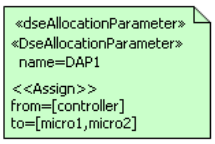


**Figure 3.** Specification of a set of architectural mappings.

Figure 3 shows an example for the EFR vocoder model where 2 feasible allocations (controller to micro1 and controller to micro2) are expressed in a single construct.

#### 2) Specification of a Space of Attribute Values

Many platform attributes can have a significant impact on system performance (cache sizes, core and bus frequencies, etc). The proposed methodology enables the association of a range of values to specific attributes of platform components for exploring their impact on performance. It is done by capturing DSE parameters, specified as UML comments associated to the component containing the explored attributes, and with the application of either a *<<DseScalarParameter>>* or a *<<DseVectorParameter>>* COMPLEX stereotype. *DSE scalar parameters* specify a sequential progression associated to a specific non-functional property. The designer can specify either minimum, maximum and step values to define possible values or annotate a specific sequence of values. *DSE vector parameters* enable modelling vector parameters with constraints on the possible combinations of the elements.

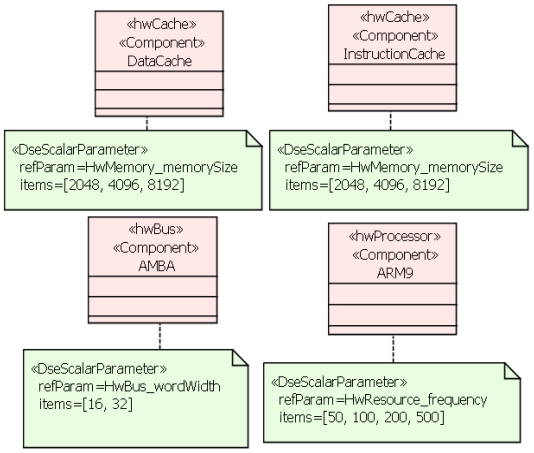


**Figure 4.** Attribute value space for the EFR Vocoder model.

Figure 4 shows the modelling of the platform parameters space for the EFR Vocoder model. This space is defined by the possible values of data and instruction cache memories (2K, 4K and 8K for ARM920T cores); by using either a 32 or 16 bits (ARM thumb instruction set) bus; and by the consideration of 4 core frequencies. This space is specified through four DSE scalar parameters related to components of the Platform view. Thus, the size of the space of platform attribute values is 4 (core frequencies) x 3 (data cache sizes) x 3 (instruction cache sizes) x 2 (data bus word) = 72 design alternatives.

#### 3) Specification of the Architecture Space

DSE parameters and *<<Assign>>* comments widen the design space without changing the platform architecture. However, the user might be interested in exploring the impact on performance of different platform architectures. The COMPLEX UML/MARTE modelling methodology supports the specification of different architectures in the same model. It is done by enabling the specification of several architectural views, in a similar way as several architectures can be associated to an entity in VHDL. This way the designer can include in the DSE loop solutions where the platform architecture is very different and thus the alternatives cannot be represented by means of a parameterized template. Additionally, it is also possible to capture a cluster of processors through a single component instance, whose multiplicity is associated to a DSE parameter.

### C. DSE Rules

The proposed modelling methodology enables delimiting the size of the design space by means of *DSE*

*rules*. It is useful since the design space can exponentially grow up to making the exploration unaffordable. Moreover, all possible solutions might not be feasible, e.g., all platforms modelled might not be available. DSE rules are specified as comments with the COMPLEX <<*DseRule*>> stereotype, and they can refer to one or more DSE parameter included in the model. DSE rules create a logical condition that in case of not being fulfilled discards the design point from the set of solutions to be explored.

### *D. Estimation Parameters*

The methodology allows stating in the model which output metrics have to feed the exploration loop. They are called estimation parameters in the sense that, in a DSE context, performance metrics are obtained by means of a performance estimation technology. There are two types of DSE estimations. DSE Application Estimations refer to non-functional attributes corresponding to the application components, e.g. the minimal inter-arrival time of a sporadic service. DSE Platform Estimations refer to the performance metrics on the platform resources, e.g. load of CPU, power consumption, etc. The estimation parameters are defined in the Architectural View by means of UML comments stereotyped with the COMPLEX <<*DseAppEstimation*>> and <<*DsePlatformEstimation*>> stereotypes.

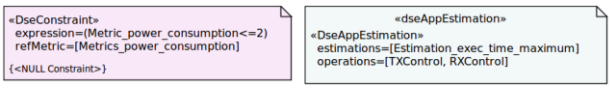


**Figure 5.** DSE constrains and estimations.

Estimation parameters, that is, output metrics to be used by the exploration tool, can be also referenced in the methodology to define DSE rules and constraints. The right hand side of Figure 5 shows how the model states the exporting of two application performance metrics of the EFR vocoder for their consideration in the exploration loop: the maximum execution time of the *TXControl* and of the *RXControl* operations (within the "controller" module). These metrics can be used for the definition of the objective function(s), which is tool dependent and managed by the COMPLEX toolset.

### *E. DSE Constraints*

DSE constraints are logic expressions referring to one or more estimation parameters (that is, output metrics). When the simulation of an exploration cycle finishes, the COMPLEX toolkit reads the output performance metrics and evaluates the logic expression to check the validity of the design point. If the logic expression is proved to be false the exploration tool discards such design point from the analysis space. Therefore, DSE constraints support from the modelling level the reduction of the set of design alternatives in real-time systems by relying on output metrics (in contrast to DSE rules, which directly bound the design space). For instance, the DSE constraint on the left hand side of Figure 5 states that the system power consumption, a platform performance metric has to be fed to the DSE loop and that any design solution involving power consumption over 2W has to be discarded.

## V. EXPERIMENTAL RESULTS

In order to show the capabilities and advantages of the proposed methodology, the EFR vocoder model previously introduced has been developed. Such a model states a space of design solutions, result of the cross product of the allocation space of Figure 3 (2 alternatives) and of the space of parameters of Figure 4, defined by the bus word width, the size of cache memories, and the frequency of the cores. Therefore, the number of solutions of the design space was:

$$N_{SOLUTIONS\ EXPLORED} = 2 \cdot 3 \cdot 3 \cdot 4 \cdot 2 = 144 \text{ solutions.}$$

The virtual system was generated with the COMPLEX tooling and a full search (that is, an exhaustive search) performed, which involved 144 SCoPE+ simulations of 1s of simulated time. The full search was totally automatic, and required no change on the model or user intervention. The exploration tool passed to the SCoPE performance model a system configuration file, and SCoPE+ returned the corresponding output metrics files.

| **Time (s)** | **real** | **usr** | **sys** | **usr+sys** |
|---|---|---|---|---|
| Full search | 559.5 | 220.1 | 171.6 | 391.7 |
| 1 simulation | 4.1s | 1.59 | 1.01 | 2.60 |

**Table 1.** Time for one simulation and for the full search.

The time required for the full search was measured and reflected in the first raw of Table 1. The time required for one simulation was measured three times and calculated its average. This was reflected in the second raw of Table 1. The "usr+sys" time is considered for the comparisons. This way the error due to the debug print outs and the load of the host system is minimized. From the results of Table 1, the average of time spent in the simulation of each alternative (reminding that the full search explored 144 solutions) was 2.72s. Assuming that the simulation of each design point does not varies much from 2.6s (this is reasonable since each iteration runs the simulation till the same simulated time, 1s), then the COMPLEX framework only spends 0.12s on average (4.4% of the simulation run) for preparing and launching the simulation of the next design alternative.

Therefore, without having exploited yet the smart search features of the exploration tool (since a full search was done), the integration and automation of the DSE loop, enabled by the proposed modelling methodology, means a significant exploration speed up with regard to any alternative which requires manual edition of the model and/or regeneration of the performance model. For instance, if the model edition, generation of the performance model, and launch of the simulation took only 2s on average, (what would be already a very fast user performance) then the full search would take 662.7s (and our proposal would yet yield 40% speed up). The speed up will be significant if the comparison is made against a methodology which requires as well the regeneration of the executable performance model. If, for instance, the user managed to edit the UML model, regenerate the executable performance model and pass the new arguments in 10 seconds on average, the full

search takes a half hour, and the speed up reaches 78.4%, that is, close 1 order of magnitude.

Figure 6 shows one of the results obtained after the exploration: a Pareto diagram which reflects the trade-off between the latency in the codification of voice subframes, and the power consumption of the system.

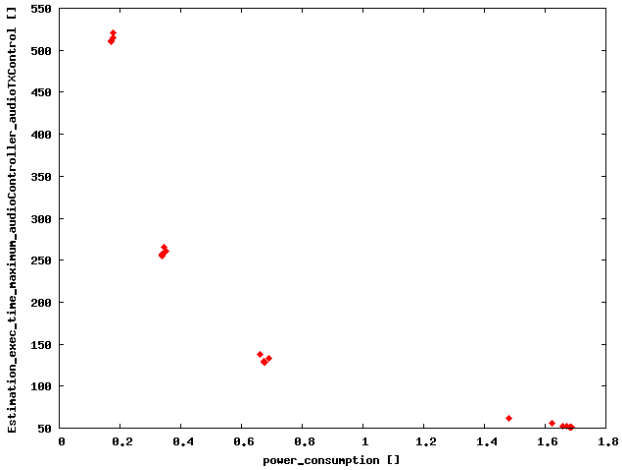


**Figure 6.** Latency vs Power Consumption in the coding.

## VI. CONCLUSIONS

This paper has presented the COMPLEX UML/MARTE modelling methodology. This methodology supports, as well as typical features of other related system-level HW/SW co-design methodology, specific features which make it specifically suited for DSE, a key activity for the design of complex embedded systems. The suitability of the proposed modelling methodology for DSE is supported by its integration in the COMPLEX flow and tooling, where the proposed UML/MARTE modelling is the starting and more user interactive stage. COMPLEX DSE methodology and tooling automates the generation of a fast executable performance model and the iterative exploration of the design space.

The proposed modelling methodology contributes to this automation since it provides the necessary concepts to define an N-dimensional design space surface and feed the generation of the configurable performance model. Moreover the methodology also enables the capture of the performance metrics to be employed in the objective functions of the DSE loop. Moreover, the model can constrain the design space, directly by means of DSE rules, and indirectly through constrains on the performance metrics. Overall system performance metrics are supported. Moreover the presented work shows how SW non-functional properties (i.e. task deadlines, minimum inter-arrival time) can be also taken into account in the DSE loop. Experimental results shows how the proposed approach enables a fully automated, and therefore a significant speed up, of the DSE phase.

## VII. ACKNOWLEDGMENTS

This work has been funded by the European FP7-247999 COMPLEX project and by the Spanish MCI TEC2011-28666-C04-02 DREAMS project.

## VIII. REFERENCES

[1] M. Holzer. Design Space Exploration for the Development of Embedded Systems. *Thesis Dissertation in the TU Vienna.* April, 2008. Vienna. Austria.

[2] Chang, H., Cooke, L., Hunt, M., Martin, G., McNelly, A. J., and Todd, L. 1999 Surviving the SOC Revolution: a Guide to Platform-Based Design. Kluwer Academic Publishers.

[3] Unified Modelling Language™,( http://www.omg.org/spec/UML/)

[4] OMG. "UML Profile for MARTE: Modelling and Analysis of Real-Time Embedded Systems". Version 1.1. Available in http://www.omgmarte.org/ . June, 2011.

[5] COMPLEX project website. http://complex.offis.de. 2012.

[6] SCoPE website. www.teisa.unican.es/scope. Dec., 2011.

[7] Multicube Explorer. http://home.dei.polimi.it/zaccaria/multicube_explorer_v1/Home.html

[8] E. Piel, R.B.Atitallah, P.Marquet, S.Meftali, S. Niar, A. Etien, J.L.Dekeyser, P. Boulet. "Gaspard2: from MARTE to SystemC Simulation". In Design, Automation and Test in Europe (DATE 08), Munich, Germany, March 2008.

[9] J.L.Dekeyser, A. Gamatié, A. Etien, R.B.Atitallah, P. Boulet. "Using the UML Profile for MARTE to MPSoC Co-Design". In First International Conference on Embedded Systems & Critical Applications (ICESCA'08), Tunis, Tunisia, May 2008.

[10] J. Vidal, F. de Lamotte, G. Gogniat, P. Soulard, J.P. Diguet. "A Code-Design Approach for Embedded System Modelling and Code Generation with UML and MARTE". In Proceedings of DATE'09. Dresden. March, 2009.

[11] A. W. Liehr, H. S. Rolfs, K. J. Buchenrieder, and U. Nageldinger. "Generating MARTE Allocation Models from Activity Threads" in "Languages for Embedded Systems and their Applications". Lecture Notes in Electrical Engineering, 2009, Volume 36, Part I, 43-56.

[12] T. Robert, V. Perrier. "COFLUENT Methodology for UML: UML SysML MARTE Flow for CoFluent Studio". White paper. Available at http://www.cofluentdesign.com/index.php/solutions/uml-sysml-marte. February, 2012.

[13] M.Mura, L.G.Murillo, M.Prevostini. "Model-based Design Space Exploration for RTES with SysML and MARTE". In proceedings of FDL'2008. Stuttgart, Germany. 2008.

[14] C. Szyperski, Component Software: Beyond Object-Oriented Programming. 2nd ed. Addison-Wesley Professional, 2002.

[15] D. C. Schmidt, "Model-driven Engineering" IEEE Computer, vol. 39 no. 2, pp. 25-31, 2006.

[16] K. Yamashita. (2010). "Possibility of ESL: A software centric system design for multicore SoC in the upstream phase", Design Automation Conference (ASP-DAC), Proc. of the 15th Asia and South Pacific. pp. 805 - 808.

[17] F. Herrera, P. Peñil, H. Posadas and E. Villar. "A Model-Driven Methodology for the Development of SystemC Executable Environments". In Proc. of FDL'2012. Vienna. Sept, 2012.

[18] Panunzio, M., Vardanega, Tullio. (2009). "On Component-Based Development and High-Integrity Real-Time Systems", Proc. of the 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications.

[19] Panunzio, M., Vardanega, Tullio. (2010). "A Component Model for On-board Software Applications", Proc. of the 36th Euromicro Conf. on Software Engineering and Advanced Applications (SEAA).