Challenges in Combining SysML and MARTE for Model-Based Design of Embedded Systems

Huascar Espinoza[1], Daniela Cancila[1], Bran Selic[2], and Sébastien Gérard[1]

[1] CEA LIST, Model-Driven Engineering Labs (LISE)
Point Courrier 94, 91191, Gif sur Yvette, France
{huascar.espinoza,daniela.cancila,sebastien.gerard}@cea.fr
[2] Malina Software Corporation
10 Blueridge Court, Nepean, Ontario, Canada
selic@acm.org

Abstract. Using model-based approaches for designing embedded systems helps abstract away unnecessary details in a manner that increases the potential for easy validation and verification, and facilitates reuse and evolution. A common practice is to use UML as the base language, possibly specialized by the so-called profiles. Despite the ever increasing number of profiles being built in many domains, there is still insufficient focus on discussing the issue of combining multiple profiles. Indeed, a single profile may not be adequate to cover all aspects required in the multidisciplinary domain of embedded systems. In this paper, we assess possible strategies for combining the SysML and MARTE profiles in a common modelling framework, while avoiding specification conflicts. We show that, despite some semantic and syntactical overlapping, the two are highly complementary for specifying embedded systems at different abstraction levels. We conclude, however, that a convergence agenda is highly desirable to align some key language features.

Keywords: model-based engineering, embedded systems, SysML, MARTE.

1 Introduction

The design of embedded systems is a complex process that depends more and more on the effective interplay of multiple disciplines, such as mechanical, electronics, and software engineering. In particular, the lack of a common design language between different disciplines hampers reasoning about system properties. The architecture of a system is particularly vulnerable to bad design choices made in the early design phases, which, unfortunately, often tend to show up later during the integration or construction phases. Designers of one part of the system may make wrong assumptions concerning some other parts resulting in increasing development costs due to long feedback cycles.

The use of models throughout the design process is gaining momentum in addressing these issues [20]. Models allow designers from different disciplines to share knowledge, facilitate design comprehension, and assess system-level trade-offs seeking higher quality and reliability. We subscribe to the view that both system design

R. Paige, A. Hartman, and A. Rensink (Eds.): ECMDA-FA 2009, LNCS 5562, pp. 98–113, 2009.

and integration will be reduced significantly by the use of a common modelling formalism, even for smaller projects. In particular, we believe that the widespread acceptance of UML (Unified Modelling Language) [16] by industry and the use of UML profiles for domain-specific expressiveness ease the challenge considerably. A profile is the mechanism standardized by the OMG for creating domain-specific modelling languages by refining the concepts of an existing standard language such as UML.

A number of UML profiles have been proposed for modelling embedded systems, both within a standardization context and as research outcomes [12]. In our work, standardization is a crucial concern since it promotes lower overall training costs and helps to reduce the risk of being dependent on a single tool vendor. We particularly focus on two standard UML profiles that cover, as a whole, a broad cross-section of the modelling capabilities required for the embedded system domain. On the one hand, SysML (Systems Modelling Language) [17] provides constructs to specify traceable requirements, structure and behaviour of system blocks, as well as a parametric formalism to specify equation-based analytical models. On the other hand, MARTE (Modelling and Analysis Real-Time and Embedded systems) [18] deals with time- and resource-constrained aspects, and includes a detailed taxonomy of hardware and software patterns along with their non-functional attributes to enable state-of-the-art quantitative analyses (e.g., performance and power consumption).

A major impediment to any kind of real-world application is that a single profile may not be sufficient to capture all aspects in the multidisciplinary domain of embedded systems. A number of industrial and research efforts have started to consider the use of both profiles in a synergistic manner, as described in more detail in Section 5, to cover as much as possible the description of embedded systems at different abstraction levels (e.g., [9] [14] [11] [1]). However, even in the standards world, different profiles may be mutually inconsistent and may overlap in ways that are not fully documented. Hence, it is essential to investigate ways of combining these two UML profiles to avoid conflicts and mismatches.

In this paper, we provide the basis for a comparison between the two profiles. The purpose is to identify some typical scenarios in which their combined usage is of relevant added value in the embedded systems domain and, to provide a convenient starting point for those interested in using both profiles in a complementary manner. One problem is that, because they are constructed for different purposes and follow different design rationales, they tend to define different syntaxes for the same modelling concepts. This issue immediately puts profile users in a dilemma when they try to exploit both profiles in the same system model. Some minimum alignment is necessary to deal with such overlaps. Consequently, another objective of this paper is precisely to encourage the SysML and MARTE standardization task forces to provide a convergence and alignment program for their respective technologies.

The remainder of the paper is organized as follows. Section 2 outlines SysML and MARTE and their respective modelling capabilities. Section 3 introduces some anticipated scenarios that combine concepts from both expressiveness domains. In Section 4, we provide some strategies to properly compare and integrate common modelling constructs. Section 5 discusses contributions and shortcomings of other attempts at combining both profiles. A short discussion and conclusions round out the paper.

2 Background

2.1 UML Profiling Capabilities

Because of the diverse nature of the disciplines needed for designing real-time and embedded system, it is clear that a single modelling language is not adequate to cover all the various concerns involved. Consequently, there has been much discussion about the suitability of UML for such domains compared to custom domain-specific modelling language designed from scratch [9]. The latter approach has the obvious advantage of enabling the definition of a language that is optimally suited to the problem at hand. At first glance, this may seem the ideal approach to modelling language definition, but closer examination reveals that it can have serious drawbacks. If each sub-domain is expressed using a specific language, there is the problem of integrating the various parts of the design so that the full system can be verified, or simply unambiguously understood. Another drawback of domain-specific languages is the availability of and support for industrial-strength tools and training for a such custom language (commercial tool vendors are rarely interested in supporting custom or low-volume languages). This can lead to significant and recurring expenses related to developing and supporting custom tools and providing training for them.

In contrast, although UML was designed to eliminate the accidental complexity stemming from gratuitous diversity, it still provides a built-in mechanism, *profiles*, for creating domain-specific modelling languages (DSML). Profiles are based on specializing the general UML concepts and semantics and can, therefore, take advantage of existing UML tools and expertise. For example, the domain-specific concept of a real-time clock can be derived from the more generic UML concept of an object, by the addition of new attributes and additional semantic constraints. It is even possible to use a domain-specific notation in lieu of the standard UML notation. This kind of reuse can mitigate or even eliminate some of the above drawbacks of custom DSMLs

Another advantage of the profile approach is that a profile can be defined as an *annotation profile*, meaning that it can be overlaid, *non-intrusively*, on an existing model to provide supplemental information and semantics not present in the original model. Such profiles can be applied to create domain-specific views and specializations of an underlying model. This is especially useful in multidisciplinary problems for the ability to capture cross-domain concerns. For example, a UML model can be annotated with information such that it can be analyzed for its schedulability characteristics by domain experts or specialized tools. At the same time (and independently of the schedulability view) a reliability engineer might overlay a reliability-specific view on that same model to determine its overall reliability characteristics. Some parts of the MARTE profile are significant examples of this profile usage.

2.2 SysML and MARTE Modelling Capabilities

SysML and MARTE consider characteristics of the embedded systems domain at different abstraction levels, architectural styles, and particularly for specific purposes or application areas. In this section, we summarize their major modelling capabilities.

SysML is a UML profile "for specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities" [17]. The so-called *Block* concept is the common conceptual entity that factorizes many different kinds of system elements such as electronic or software components, mechanical parts, information units, and whatever structural entity composing the system under interest. Blocks articulate a set of modelling perspectives enabling separation of concerns during systems design. Eschewing excessive detail[1], we identify the following key contributions of SysML regarding UML:

- *Architecture Organization:* These include modelling concepts to organize system architecture descriptions as defined by the IEEE 1471 standard [6]. Among them, the concepts of *view*, *viewpoint*, and *rationale* are most important.
- *Blocks and Flows:* Block description and internal block diagrams of SysML enable the specification of more generic interactions and phenomena than those existing just in software systems. This includes physical flows such as liquids, energy, or electrical flows. The dimension and measurement units of the flowing physical quantities can be explicitly defined.
- *Behaviour:* Although most behaviour constructs in SysML are similar to UML (interactions, state machines, activities, and use cases), SysML refines some of them for modelling continuous systems and probabilities in activity diagrams.
- *Requirements:* SysML provides an explicit facility for modelling system requirements, along with their traceability with regard to the architecture evolution. These can be specified in either graphical or tabular format.
- *Parametrics:* A perspective called parametric diagram allows SysML users to describe, in a graphical manner, analytical relationships and constraints, such as those described by mathematical equations. Parametric diagrams provide a mechanism for integrating SysML design models with engineering analysis.

MARTE is a UML profile that supports specification of real-time and embedded systems [18]. In addition to functional design, this profile adds constructs to describe the hardware and software (e.g., OS services) resources and defines specific properties to enable designers to perform timing and power consumption analysis. With regard to UML, MARTE adds the following features[2]:

- *NFPs.* The NFPs (Non-Functional Properties) modelling framework provides means to specify semantically well-formed non-functional properties (e.g., throughputs, bandwidths, delays, memory usage), supported by a language to formulate algebraic and time expressions.
- *Time.* A highly refined model of time and timing mechanisms integrates concepts from different sub-domains in embedded systems design, such as causal time, synchronous time, and chronometric time.
- *Software application.* A common model of computation provides semantic support for the real-time object paradigm. This paradigm allows specifying

[1] Further information on SysML can be found via http://www.omgsysml.org/
[2] Further information on MARTE can be found via http://www.omgmarte.org/

applications at a high abstraction level, by delegating concurrency, communication, and time-constraint aspects to a modular unit called *real-time unit* (RtUnit).

- *Components.* The MARTE component model extends UML composite structures and SysML internal block diagrams with a notion of message-based communications. This is intended to support the request-reply/publish-consume communication paradigm.
- *HW/SW Resources.* Software and hardware resources can be described at different levels of abstraction, including their typical services, as found in common OS platforms, and common non-functional properties like power consumption or memory usage.
- *Quantitative Analysis.* A set of pre-defined non-functional annotations enable MARTE models to bridge with state-of-the-art performance and scheduling analysis tools.

3 Scenarios of Combined Usage

To focus our study, we identify a set of representative scenarios in which a combined usage of SysML and MARTE is of relevant added value in the embedded systems domain. Although this set is certainly incomplete, it allows us to drive our comparison in a more focused manner. The intent is to adequately answer the question of what can each profile target best in modelling, and then determine their integration issues.

3.1 Defining Architecture Frameworks

The modelling capabilities of both SysML and MARTE are rich enough for a wide range of design approaches. This has the flexibility for supporting and integrating multiple design perspectives, but also the difficulty of understanding and choosing among a variety of language alternatives. In both cases, there is not a predetermined approach to use the language constructs through the development lifecycle. This means that a consistent modelling framework and methodology should be defined for using these profiles in a particular application domain.

Architecture Organization. In the IEEE 1471 standard (and in the draft of its upcoming update ISO/IEC 42010), the concept of modelling framework is referred as *architecture framework*. An architecture framework "establishes a common practice for creating, organizing, interpreting and analyzing architectural descriptions used within a particular domain of application or stakeholder community" [6]. An architecture framework identifies one or more predefined architectural viewpoints. Viewpoints define how to construct views, which are in turn a representation of a system from the perspective of a set of modeling concerns.

SysML implements IEEE 1471 by providing a set of constructs to organize models. In particular, SysML does not define any specific viewpoint, but it provides means to specify how views are built, and to relate any user-specific view to a given viewpoint. This is aligned with the IEEE 1471 approach that envisages libraries of viewpoints, in order to enable architects selecting those useful for system design at hand.

Although MARTE does not provide any concrete model element to define viewpoints, it has an implicit conception of viewpoints rooted in its design rationale. Indeed, some of the MARTE constructs have been designed to define domain-specific viewpoints (see Section 2.1). Such viewpoints, when applied to a standard UML model, cast that model in a domain-specific way and may also add supplementary information to the model relevant to the viewpoint.

In consequence, there is no language overlapping in this respect. SysML and MARTE can be used in complementary way. While SysML provides means to create viewpoints in a general way, MARTE provides particular viewpoints. However, an open issue is to enable designers of architecture frameworks to build consistent interview rules that ensure meaningful and correct-by-construction models.

3.2 Requirements Engineering

System Usage Scenarios. Requirements engineering is the process by which the requirements for systems and software products are gathered, analyzed, documented, and managed throughout the development life cycle. UML has traditionally been used to document user requirements by means of use case diagrams. Use cases follow a graphical, scenario-based approach. This means that requirements are organized into system usage histories, acting as a user-friendly bridge between technical and business stakeholders.

Although use cases may be formalized to certain degree, for example by using sequence diagrams in order to detail such usage histories, they are often criticized for a number of limitations. For instance, use cases lack well-defined semantics, which may lead to differences in interpretations by stakeholders [22]. They are applied mainly to model functional requirements, but are not very helpful to model non-functional ones. Also, relationships between requirements and the various architectural parts that satisfy those requirements are difficult to trace. SysML and MARTE provide some significant enhancements in these aspects.

Requirements Management/Traceability. SysML requirements diagrams explicitly show the various kinds of relationships between different requirements. This enlarges the spectrum of requirements engineering tools that can interact with UML tools. In effect, the SysML requirements modelling constructs are intended to provide an automated bridge between architectural models and traditional requirements management tools such as, for instance, Requisite Pro, Rectify, or DOORS [1]. The latter provide support for traceability analysis, flow-down, derivation, assignment, among other requirement engineering activities. In particular, requirements tracing is very useful, for example, to identify how requirements are affected by changes, and to prioritize requirements. Traceability also provides a possibility of verifying whether or not all requirements have been fulfilled by the system and sub-system components.

Non-Functional Requirements. On its side, MARTE offers key features to specify non-functional requirements in general and timing requirements in particular. In embedded systems development, non-functional characteristics (e.g., performance, reliability, power consumption) influence a wide range of design decisions [3]. One possible scenario is using MARTE annotations to characterize non-functional constraints in use case diagrams and their underlying sequence diagrams. This provides two important capabilities leading toward more formal requirements specification.

First, non-functional requirements are cohesively specified along with functional requirements. While specifying non-functional aspects is possible with SysML requirements diagrams, their semantic relationship to concrete functional system usages is hard to capture. In particular, the completeness of requirements satisfaction in real-time systems is strongly dependent on the coupling between system function and timing. In MARTE, timing annotations provide semantic definitions closely related to the system behavior. For instance, one may define a jitter constraint in the arrival of an event and identify if such event relates either to a send, receive, or consume occurrence within a sequence diagram. Second, non-functional annotations follow a well-defined textual syntax, which is supported by the MARTE's Value Specification Language (VSL). The main advantages of this level of formalization are the ability to support automated validation, verification, traceability, and, more simply, an unambiguous understanding by stakeholders.

Clearly, SysML and MARTE concepts, articulated by use cases and scenarios, are highly complementary. While scenarios are useful for managing change and evolution, managing scenario traceability across multiple changes becomes increasingly difficult. SysML contributes with constructs to define such traceability relations. Additionally, MARTE completes scenario precision with well-formed non-functional annotations. However, it is important to define clear consistency rules to combine them in a typical development process using different requirements engineering tools.

3.3 System-Level Design Integration

In a typical development process for embedded systems, software and other forms of engineering will be at least partially concurrent. The system is developed by composing pieces that, all or in part, have already been pre-designed or designed independently by different teams specialized in different disciplines. This is often done in vertical design chains such as, for example, in the avionics and automotive industries. Therefore, there is a need for supporting design artefacts by common and standard specification formalisms that will allow plug-and-play of subsystems and their implementation [23].

A model view is a typical abstraction that helps to divide a complex problem into smaller and comprehensible parts. In order to integrate global models, e.g., for performing system-level analysis, we must recombine these smaller parts in a consistent way. UML supports model composition by means of composite structure diagrams. The basic principle is to define usages of model elements in a given context. The idea of composite system models is to describe how information from multiple modelling artefacts and views is to be joined, deployed, or configured. Although there is a linguistic divergence[3], both SysML and MARTE reuse this notion with some particularities. Thus, some aspects need to be taken into consideration for their combined use.

Hierarchy and Composition. To understand the pragmatic problems of SysML-MARTE joint usage, let us consider the scenario of a large development project with engineers from multiple disciplines. It should be carefully decided how the system model will be created by integrating the models from different disciplines. One important issue is the layering and mismatched sub-system hierarchies, which has been

[3] While SysML uses the term "block" for such composition units, MARTE uses "structured component".

comprehensively addressed by Maier [15]. For instance, in multiprocessor software-intensive design, the electronic system perspective typically represents a hierarchy of interconnected processors, each containing software units. From the software perspective, the hierarchy is reversed, as generally illustrated in MARTE examples [18]. At the top is a distributed application, composed of software units that interact through data- and message-based interfaces. Below the application are the operating system (OS) and library layers that support the distributed application. At the bottom of the hierarchy, the hardware (processors and networks) completes the model.

This aspect is important when deciding which kind of modelling constructs will be used to represent hierarchy, allocation/deployment, and composition. For instance, while a composition relationship would be used for the hardware viewpoint, an allocation relationship (supported by both SysML and MARTE) would be preferred by the software designers. Some compatibility or merging rules need to be defined to provide system-level consistency.

While in some cases, engineers from a given discipline would exclusively use either SysML or MARTE, in other cases they would need to combine concepts from both profiles. An integration scenario may consist in starting from a system-level model, probably specified with SysML blocks, and in adding later some additional semantics to some of these blocks, for example by applying MARTE stereotypes. Indeed, the detail level underlying MARTE constructs makes possible to specify some aspects such as concurrency and synchronization mechanisms, as well as resource patterns such as processing resources, communication buses, or power supply devices along with a set of predefined quality attributes. This is especially required in application areas where designers are interested in preparing models to perform simulation, quantitative analysis or product synthesis.

Interfacing/Interaction. A central concern in system and software architecting is to understand the interfaces and interactions between structural elements. The nature of such interfaces and interactions can significantly vary from software to other kind of systems. Looking at the structural aspects, we can see that MARTE adopted the notions of *port* and *flow* from SysML. This may seem very convenient from a perspective of semantic consistency. However, SysML flow ports require careful attention when used to model flows of physical quantities, such as for example energy or torque. Cares must be exercised in defining explicit behaviour on flow transmission. SysML physical flows are often continuous in time, whereas MARTE flows are used to describe data transmission with particular delegation semantics. While providing a precise semantics to flows is currently outside the scope of both profiles, their combined use should define a common "semantic envelope" that could be shared by SysML and MARTE. In this way, composing models from different disciplines will preserve system-level consistency.

3.4 Engineering/Quantitative Analysis

Engineering analysis (SysML term) or quantitative analysis (MARTE term) concern the use of mathematical techniques to study certain quality attributes of the system. They include stress, thermal or fluid analysis in mechanical engineering, and performance or reliability analysis in software engineering. One challenging problem in model-based engineering is to integrate models that are commonly used for system

production or software code generation with the information that is relevant to perform analysis [7]. The goal is to reduce the time required to prepare a design model for performing analysis and to ensure greater accuracy of an analysis model by directly associating it with the actual system model. Both SysML and MARTE provide key contributions in this direction, but some alignment work has to still be done.

Timing Modelling. Beyond the annotation of quality attributes, timing analysis requires a careful semantic definition closely related to the system behavior and the different models of computation and communication [2]. SysML does not extend the UML time model, but a set of preliminary requirements were established by its standardization board, including continuous time models and relativistic effects that can occur in distributed systems. In MARTE, time modelling is a core concern. We can distinguish at least three layers of time constructs:

- In a first layer, time is presented as a set of fundamental notions such as time instant, duration, time bases, or clocks. These provide an unambiguous basis to express further modeling constructs and well-formed value spaces for data types.
- In a second layer, MARTE provides mechanisms to annotate timing requirements and constraints in UML models. One key modeling feature is the concept of *observation*. Observations provide marking points in UML models to specify assertions. Some typical assertions have been embedded in ready-to-use patterns, such as for example jitters.
- In the third layer, time concepts are defined as part of the behavior, not mere annotations. This set of constructs cover both physical and logical time. While the logical time is the basis to understand basic temporal notions, this is further refined to support precedence/dependency in presence of concurrency, and clocked time abstractions to cover synchronous language abstractions (such as those from Lustre, Signal or Esterel).

While the adoption of the two basic layers is certainly useful for system engineering in general, the third layer would need some extensions to include, for example, modelling of the continuous dynamics of systems [12]. This would need to provide means to specify system behaviour in terms of hybrid discrete event and differential algebraic equation systems.

Quantities Values. In SysML, a value property represents a quantifiable characteristic of a block (e.g. energy consumption, surface, and temperature range of a microprocessor). Value properties are defined in block compartments by assigning a name and a value type. A value type is a kind of data type that carries a particular pair consisting of a dimension and a measurement unit.

For its part, MARTE uses its Non-Functional Properties (NFPs) modelling framework. The NFPs modelling framework provides the ability to encapsulate rich annotations within non-functional values. For instance, consider a property named ``latency". Instead of specifying its meaning in an axiomatic way such as: "duration in milliseconds with an accuracy of 0.01 measured by simulation as a mean value", the specification itself include all this information in a normalized syntax. For this purpose, the MARTE data type system includes the required data structure (value, unit, precision, measurement source, etc.) in a predefined library. For example, Duration, Frequency

and Power are typical non-functional data types. Different units of the same physical quantity may be transformed to, or expressed in terms of, existing base units through a given conversion factor and an offset factor.

One of the main issues when trying to combine both profiles is that the modelling approaches to declare and specify quantitative values is quite different. The main difference is that SysML hard coded the qualification of value types with the stereotypes unit and dimension, while MARTE allows for declaring a set of qualifiers as an extendable library. As a consequence, using both modelling mechanisms in the same model may lead to inconsistencies and cumbersome model processing. Alignment of these two modelling styles is a key issue that is being dealt as a joint effort between the MARTE and SysML task forces at OMG.

Beyond syntactical issues, the debate should be centred on providing practical capabilities to both profiles. We believe that at least two key capabilities should be allowed from SysML and MARTE models:

- *Measurement conversion*. Quantities need to be expressed in different measurement units while still allowing tools to convert quantities from one set of units to another.
- *Dimensional analysis*. Physical expressions must guarantee the consistency of equations and solve resulting measurement units and dimensions.

If we look at SysML, it forces tools to be hard-coded with the transformations between measurement units (e.g., from "mm" to "m") because unit definition lacks conversion factors. Furthermore, dimensional analysis is not possible in SysML since dimensions are not defined in terms of basic dimensions and their exponents (e.g., $F = LMT^{-2}$). Conversely, while MARTE supports unit conversion, the notion of dimension has not been considered at all.

Parameters/Expressions. Parameterized expressions are a primary feature in order to prepare models for analyzing performance, risk, costs, and so on [7]. SysML parametric diagrams capture constraints among performance, physical, and other quality-related properties of the system and its environment. Such constraints are specified as equations among value properties. Equations can be specified in a third-party language (e.g., MathML or Modelica). The basic composite modelling entity is the Constraint Block. The relationships between modelling entities within a constraint block are not committed to an 'input' or 'output' role early. Thus, they are called non-causal, as opposed to data flow and control flow approaches. Non-causal models are suitable to enable analytic processing, and can increase the level of integration/automation between design tools and analysis tools.

In addition, MARTE's VSL gives the syntax to formulate algebraic and time expressions. VSL is rooted in OCL. However, VSL was intended to provide more compact expressions. In addition, VSL extends arithmetic and logical expressions with time-related annotations, which can be extended by libraries providing new functions.

We believe that a combined use of SysML parametric diagrams and VSL would provide significant advantages. While parametric diagrams provide a user-friendly formalism to specify non-causal models, VSL provides the textual syntax for constraint expressions. One open issue in VSL is its extension to support special expressions used in system engineering. For instance, differential and integrals, continuous time expressions, and discrete event equations.

4 Combination Strategies

In this section, we outline some issues in combining SysML and MARTE and propose general strategies to integrate both profiles in a single modelling framework. Table 1 summarizes the modelling aspects discussed in Section 3 along with a set of profile combination cases and implementation issues, which are elaborated below.

Table 1. MARTE/SysML Combination Issues

Modelling Concern (from Section 3)	SysML concepts (examples)	MARTE concepts (examples)	Conflicting Combination Cases*	Implementation Issues*
Architecture Organization	view, viewpoint, rationale,…	-	-	predefine library of MARTE viewpoints
Hierarchy/Composition	block, part, allocation	structural component, parts, hw/sw patterns, application-platform allocations	(a)	(1) & (2)
Interfacing/Interaction	port, flow, items	idem SysML + message-based	(a) (c)	(2)
Spectrum of Behavioral Models	rate, continuous, discrete edges, probability,…,	synchronous/asynchronous, causal/real-time	(c), (d)	(1)
System Usage Scenarios	use case, sequence diagrams	use cases, sequence diagrams	common UML concepts	-
Requirements Processing/Trace	requirement, trace relationships, test case	-	-	(1) SysML requirements can be fully imported
Non-Functional Requirements	requirement	nfp constraint, VSL expressions	complementary	(1)
Time Modelling	(UML) time constraints	extended time constraints, clocks, predefined nfp's for time analysis	-	(1) & (2) not all MARTE time notions required
Quantity Values	value property, value type, unit, dimension	nfp, nfp type, unit	(c) (d) overlapping	(2) language alignment required
Parameters/Expressions	constraint blocks, parametric diagrams	VSL expressions	(c) complementary	(1) VSL as expression language

* see text for full explanations

Combination Case. We can generalize typical categories of the combined usage of UML profiles (this is applicable for two or more profiles) as follows:

a) Each language is used for different partitions of the system, in which case they are practically mutually exclusive and conflicts are small or even negligible. For example, SysML is used for mechanical design and MARTE for software design. As shown in Table 1, this category needs special attention when defining the hierarchy/composition and interfacing/interaction constructs during a system-level integration phase.
b) Each language is used for a different level of abstraction. Again, there is not much conflict here. For instance, SysML is used for system domain analysis and MARTE for a detailed design.
c) The languages are used in combination into the same parts of a model (e.g., in the same modelling view) and for the same purpose or concern. For instance, we may use the SysML facilities for continuous behaviour in activity diagrams and the MARTE time annotations to support performance analysis.

d) The languages are used in combination but for different purposes such as, for example, using MARTE annotations to do performance analysis on a SysML model. The UML profiling capabilities of being able to apply many stereotypes to a single model entity is crucial for this kind of usage. There may be some conflict in trying to keep the consistency between MARTE non-functional annotations embedded in stereotype attributes (e.g., performance analysis stereotypes), with other SysML specifications such as block quantity value annotations or block constraint parameters.

Implementation Issues. The above combined cases may result in different combination issues from a tool implementation viewpoint. A supporting toolset that accompanies UML profiles is, strictly speaking, not a part of the language problem. However, the utility of a profile combination is directly related to the maturity of the supporting tools. We identify the following scenarios in combining MARTE and SysML profiles in modelling tools:

1) The simplest solution is to apply the profiles (i.e., the full profile definition) or sub-profiles (i.e., sub-packages stereotyped as profiles) where needed within a model. For example, a SysML user could specify that it requires the full Time Modeling package of MARTE. UML tools can manage this case because of the modularity defined in MARTE (organized in "extension units") and the UML's ability to select only those profile packages that are of direct interest.
2) While one may likely use some concepts of a profile or sub-profile, designers may not want to include the full profile or sub-profile package in their models. For instance, MARTE profile users may want to gain access to SysML concepts of block, but they may prefer to use the MARTE constructs for flows. UML does not allow for applying single stereotypes (contained in a profile) into a model. What is needed is a decoupling/merging mechanism to compose profile concepts and to make it available for profile users. Managing semantic compatibility is a requirement here.

In general, a hypothetical MARTE-SysML modelling tool should allow for filtering appropriate information according to specific users. Some engineering disciplines may be satisfied with a high-level description (e.g., blocks-and-flows description), software developers may want detailed behaviour specifications, while analysis experts may require information of non-functional properties. This aspect is more relevant when more than one stereotype is applied to a single UML model element. For instance, one may consider a SysML Block, as a specific hardware resource by annotating it with the appropriate MARTE stereotype. However, it is considered as a "resource" from a software viewpoint, but not from an electronic viewpoint. This needs a suitable presentation mechanism to show the right stereotype to various stakeholders.

4.2 Combination Clues

Defining a modelling framework that combines SysML and MARTE requires a systematic comparison of the two. We consider that at least the following aspects should be assessed in such work:

1. *Conceptual Domain Coverage*. Beyond syntactical aspects, it is important to begin by assessing both profiles from a conceptual viewpoint. The intent is to

reach an overall understanding of these profiles and determine what application domains are best covered by each. A good starting point is using the conceptual domain models underlying the UML profiles. Conceptual domain models are created as free as possible from considerations related to specific solution technologies so as to not embody any premature decisions that may hamper later language use. Currently, the MARTE specification provides a conceptual domain model in the form of a metamodel with a textual description. On the other hand, SysML directly defined UML stereotypes extending the UML metamodel. Although a conceptual description is provided, a metamodel would significantly help on identifying/comparing conceptualization entities of the targeted domain.

2. *Semantic/Syntactic Overlapping*. The evaluation of related points between both profiles should be clearly identified by defining overlapping semantics (conceptual coverage), abstract syntax (extended UML constructs), and concrete syntax (symbols and terminology). The intent is to determine which aspects of both profiles can be consistently aligned and/or selected to consistently use both profiles. Overlapping aspects must be assessed in the light of one of the language use cases, (c) or (d), identified in Section 3.1. While case (d) needs revisiting the notion of views and viewpoints in the context of UML profiles (see Section 2.1), case (c) requires a more careful treatment of semantic consistency.
3. *Usability/Pragmatics*. Usability issues are concerned with such concepts as ease of use, productivity, and user satisfaction. Once the overlapping concepts are identified and before deciding which profile features to adopt in a given modeling framework, we should identify the effectiveness of different symbols or stereotype names for model understandability, as well as the number of steps needed to accomplish a modeling goal. Of course that may depend on a tools' maturity. However, syntactical design choices can help avoid complicated ways of performing modeling steps or features which invite mistakes.
4. *Expressiveness Limitations*. One fundamental requirement that should drive a useful comparison is completeness and lack of model expressiveness. The evaluation of missing aspects needs to be objective by clearly identifying whether it implies a conceptual, semantic, or attributes insufficiency. This raises the problems of improving and extending both profiles, which is an important goal of our research.
5. *Abstraction/Refinement Levels*. One fundamental difference between SysML and MARTE relates to their ontological considerations. For example, while SysML does not consider any "functional" classification of structural elements (only the generic concept of Block exists), MARTE goes deeper by providing a detailed taxonomy of application and resource structural elements. Using abstract or concrete language concepts will depend on the phase of development, and the kind of model processing (simulation, verification, etc.) required at each level.

5 Related Work

The academic and industrial communities have recently begun to investigate the complementary use of SysML and MARTE to support model-based development of embedded systems.

Among current projects in the embedded systems domain, MeMVaTEx [1] defines a model-based methodology for modelling, validating, and tracing system requirements. It relies on SysML for requirements modelling and on MARTE for modelling timing aspects. Since these aspects are practically independent, their combination is handled methodologically, by providing consistent rules on when and where to apply concepts of the individual profiles. Another project combining these two profiles is INTERESTED [11], which attempts to create an interoperable tool-chain for enhanced rapid design and prototyping of embedded systems. This work aims at a more extensive use of SysML and MARTE. While the first profile serves to describe the high-level architecture organized around functional blocks, the second one provides the standard annotations to enable timing analysis. However, the methodological rules to guide the combined use of both profiles have yet to be established.

Two additional projects were recently started with the objective of adopting SysML and MARTE in the hardware/software co-design field. One of these, the SATURN project [19], proposes to bridge the gap between SysML/MARTE modelling and tools for architecture exploration, simulation and synthesis (in SystemC/VHDL for hardware and C/C++ for embedded software). The main strategy is to adopt most of the constructs of SysML and to integrate MARTE for adding the formal semantics of different models of computation and thus enable system verification. The second project, Lambda [14], intends to reconcile a number of related standards, including SysML, MARTE, AADL, and IP-XACT, to develop a library of broadly used software and hardware platforms.

At the other end of the spectrum, there is very little research literature discussing integrated approaches for system and software modelling based on UML. An example is [10], where the authors evaluate how UML and SysML could be consistently used for both system and software modelling. Perhaps, the main contribution of this work is a mapping between SysML and UML concepts and the identification of the application domains associated with each concept. Unlike this work, we attempt to provide a more rigorous comparison of system and software modelling concerns, and additionally, enrich expressiveness with MARTE features.

With regard to the combination of profiles at tooling level, the authors in [4] introduce a packaging unit called MDATC (which stands for Model-Driven Architecture Tool Component) that serves to collect metamodels and/or profiles, know-how, and required resources in order to support domain-specific activities. Thus, by using MDATC, modelling rules and constraints in the use of multiple profiles can be represented and exchanged in a standard format.

We end this section by highlighting the general problem of composition of languages or profiles. For instance, an aspect oriented approach supporting metamodel composition is proposed in [8]. The authors focus on implementing composition mechanisms for matching and merging model elements that crosscut the dominant structure described in a primary model. The composition directives are implemented in Kermeta, an open-source metamodelling language. Even if language composition between different metamodels is certainly a more difficult problem than combining stereotypes extending the same metamodel, especial care must be exercised. Our study can be inserted in this lively context and viewed as a modest contribution in composition of profiles, with special focus on SysML and MARTE, although in general fragmentation problem is left as an open problem.

6 Conclusions

Because of the varying nature of the disciplines involved in embedded system design, it is clear that a single modelling language, such as for example UML, may not be suitable for all aspects. We believe that the UML profile mechanism is well suited to create domain-specific languages, by providing a common semantic and syntactic foundation while also permitting reuse of the underlying modelling tools. Currently, there are an important number of profiles that may make their usage cumbersome, as they are often created mutually inconsistent and overlapping. In this paper we presented some integration strategies for combining the SysML and MARTE profiles. Both provide essential ingredients to model embedded systems. Our intent is to offer a better understanding of their conceptual domains, and to help in using both profiles in a single model by avoiding semantic and syntactical mismatches.

We presented some typical scenarios in which their combined usage is of relevant added value in the embedded systems domain. In general, using modelling constructs from one or the other profile depends on the expressive power a constructs should provide to practitioners. In a simple usage scenario, the intent may be to aid understanding and to communicate about a system design. As such, it is not necessary to define a detailed description or precise semantics, and basic evaluations of the architecture could be performed. In a more elaborated scenario, however, we may be interested on using powerful analysis tools, simulators, model checkers, product synthesis tools, and the like. In this case, the necessary levels of specification detail and semantic precision are much higher. While both forms of specification have merit, their usage will be driven by the specific needs of a particular development process and its phases through the system lifecycle.

Our future work consists in providing a detailed comparison of SysML and MARTE's semantic and syntax, providing pertinent examples on their combined usage, and suggesting some improvements regarding language mismatches.

Acknowledgments. The work presented here is partially carried out within the System@tic competitiveness cluster projects Lambda and IMOFIS.

References

[1] Albinet, A., Begoc, S., Boulanger, J.-L., Casse, O., Dal, I., Dubois, H., Lakhal, F., Louar, D., Peraldi-Frati, M.-A., Sorel, Y., Van., Q.-D.: The MeMVaTEx methodology: from requirements to models in automotive application design. In: 4th European Congress ERTS Embedded Real Time Software. Toulouse, France (January 2008)

[2] André, C.: Time Modeling in MARTE. In: FDL 2007 Forum on specification and Design Languages, Barcelona, Spain (2007)

[3] Cancila, D., Passerone, R.: Functional and structural properties in the Model-Driven Engineering approach. In: ETFA 2008 (2008)

[4] Bendraou, R., Desfray, P., Gervais, M.-P., Muller, A.: MDA Tool Components: a proposal for packaging know-how in model driven development. Software and System Modeling 7, 329–343 (2008)

[5] Cuccuru, A., Gérard, S., Radermacher, A.: Meaningful Composite Structures - On the Semantics of Ports in UML2. In: Czarnecki, K., Ober, I., Bruel, J.-M., Uhl, A., Völter, M. (eds.) MODELS 2008. LNCS, vol. 5301. Springer, Heidelberg (2008)

[6] Emery, D., Hilliard, R.: Updating IEEE 1471: architecture frameworks and other topics. In: Seventh Working IEEE/IFIP Conference on Software Architecture WICSA (2008)
[7] Espinoza, H., Servat, D., Gérard, S.: Leveraging Analysis-Aided Design Decision Knowledge in UML-Based Development of Embedded Systems. In: SHARK at ICSE 2008, Leipzig (May 2008)
[8] France, R., Fleurey, F., Reddy, R., Baudry, B., Ghosh, S.: Providing Support for Model Composition in Metamodels. In: Proceedings of EDOC 2007, Annapolis, USA (October 2007)
[9] Gray, J., Tolvanen, J.-P., Kelly, S., Gokhale, A., Neema, S., Sprinkle, J.: Domain-Specific Modeling. In: CRC Handbook of Dynamic System Modeling. CRC Press, Boca Raton (2007)
[10] Hause, M., Thom, F.: Building Bridges Between Systems and Software with SysML and UML. In: INCOSE Intl. Symposium (June 2008)
[11] INTERESTED EU Project: Interoperable embedded systems Tool-chain for enhanced rapid design, prototyping and code generation, http://www.interested-ip.eu/index.html
[12] Johnson, T., Jobe, J., Paredis, C., Burkhart, R.: Modeling Continuous System Dynamics in SysML. In: Proceedings of the IMECE 2007 (November 2007)
[13] Lagarde, F., Espinoza, H., Terrier, F., André, C., Gérard, S.: Leveraging Patterns on Domain Models to Improve UML Profile Definition. In: Fiadeiro, J.L., Inverardi, P. (eds.) FASE 2008. LNCS, vol. 4961, pp. 116–130. Springer, Heidelberg (2008)
[14] Lambda Project, Lambda Libraries for Applying Model Based Development Approaches, Technical Annex (May 2008)
[15] Maier, M.: System and Software Architecture Reconciliation. Systems Engineering Journal, 146–159 (2006)
[16] OMG, Unified Modeling Language, UML™ Superstructure, V2.1.2
[17] OMG, Systems Modeling Language SysML™, V1.0
[18] OMG, UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded systems, Beta 2
[19] SATURN Project: SysML bAsed modeling, architecTUre exploRation, simulation and syNthesis for complex embedded systems, http://www.saturnsysml.eu
[20] Selic, B.: From Model-Driven Development to Model-Driven Engineering. In: Keynote talk at ECRTS 2007 (July 2007)
[21] Selic, B.: A Systematic Approach to Domain-Specific Language Design Using UML. In: ISORC 2007, pp. 2–9 (2007)
[22] Soares, M.S., Vrancken, J.L.M.: A Proposed Extension to the SysML Requirements diagram. In: IASTED International Conference on Software Engineering, Austria (2008)
[23] Sifakis, J.: Embedded Systems - Challenges and Work Directions. In: Higashino, T. (ed.) OPODIS 2004. LNCS, vol. 3544, pp. 184–185. Springer, Heidelberg (2005)