Softw Syst Model (2018) 17:343–361
https://doi.org/10.1007/s10270-016-0525-1

REGULAR PAPER

Model-based requirements specification of real-time systems with UML, SysML and MARTE

Fabíola Gonçalves C. Ribeiro[1] · Carlos E. Pereira[2] · Achim Rettberg[3] · Michel S. Soares[4]

Received: 25 September 2015 / Revised: 5 February 2016 / Accepted: 13 March 2016 / Published online: 7 April 2016

Abstract Activities of specification, analysis and design of real-time systems (RTS) are highly dependent on an effective understanding of the application domain and on the thorough representation of their basic requirements. Model-based approaches using modeling languages such as UML are often applied to contribute to handle complexity of RTS development. However, UML alone does not completely represent important features associated with these systems, such as relationship with hardware elements and an effective representation of timing constraints. This article explores the combined use of UML and its profiles SysML and MARTE for modeling hardware and software requirements of RTS, with application to a case of controlling urban road traffic. The SysML profile alone does not present the representation of temporal, behavioral and performance requirements. The MARTE profile provides key resources to specify non-functional requirements for RTS, in addition to a clear description of the various relevant aspects of requirements definition of RTS, as for instance, temporal aspects and constraints. The main objective is to present the combined application of SysML with MARTE stereotypes, which enables the specification of different features of individual software requirements. Thus, in addition to the factors mentioned above, we can say that the proposed approach has an important role to specify RTS at different levels of detail and levels of abstraction.

Keywords Real-time systems · Requirements engineering · UML · SysML · MARTE

Communicated by Prof. Jean-Michel Bruel.

✉ Michel S. Soares
mics.soares@gmail.com

Fabíola Gonçalves C. Ribeiro
fabiola.ribeiro@ifgoiano.edu.br

Carlos E. Pereira
cpereira@ece.ufrgs.br

Achim Rettberg
achim.rettberg@iess.org

1 Federal Institute Goiano, Catalão, Brazil

2 Federal University of Rio Grande do Sul, Porto Alegre, Brazil

3 Carl von Ossietzky Universitt Oldenburg, Oldenburg, Germany

4 Federal University of Sergipe, São Cristóvão, Brazil

1 Introduction

Requirements for a system are a collection of needs expressed by stakeholders respecting some constraints under which the system must operate. Requirements engineering is the process by which requirements for systems and software are elicited, documented, analyzed and managed throughout the development life cycle [33], including activities of tracing requirements. Requirements engineering can be divided into two main groups of activities [32]. First group is the development of requirements, which includes eliciting, documenting, analyzing and validating requirements. Second group is requirements management, which includes tracing and change management of requirements.

Among many other modeling languages, UML [31] has traditionally been applied to document requirements through Use Case diagrams [46,50]. Requirements are then organized into stories of using the system that acts as an organization between users, technical and business stakeholders [18,48] and [17]. There are some issues involved with using Use Case diagrams for modeling real-time requirements. Use Case diagrams are most often used to describe scenarios of requirements, not individual requirements. Besides, Use Cases are informal and can be misinterpreted when too many details are included in the models [1,43]. Proposals to address problems of UML in relation to modeling real-time software were introduced, including new UML profiles which add elements to model time requirements, system requirements and non-functional properties. These include MARTE [29] and SysML [30].

SysML Requirements diagram shows explicitly relationships between requirements, increasing the spectrum of understanding and defining real-time system requirements. However, the SysML profile by itself does not provide concepts for representing temporal, behavioral and performance requirements, nor provide elements for explicit representation of system configurations. MARTE profile provides key resources to specify non-functional requirements for real-time systems, generally time requirements.

MARTE and SysML profiles have been applied in past years in various domains, such as railway control [5], concurrent systems [40], design of distributed embedded systems [12], and in many other industrial environments [20,34] and [21]. However, few approaches were proposed with focus on applying MARTE and SysML together to design RTS, and even less with focus on requirements modeling, which is the purpose of this article.

One of these approaches [35] defined a systems modeling language based on subsets of MARTE and SysML, allowing iterative refinement from high-level specification to its implementation. Another approach [36] does not have focus on requirements. In other approach, described in [14], the authors assess possible strategies for combining SysML and MARTE in a common modeling framework, while avoiding specification conflicts. Although the authors came to the conclusion that the two are highly complementary for specifying embedded systems at different abstraction levels, they also found that a convergence agenda is highly desirable to align some key language features.

The work presented in [26] proposes a model-driven approach for requirement engineering targeted to the embedded software domain, named MDEReq (Model-Driven Engineering for Requirement Management). The approach integrates UML, MARTE and SysML models to improve and facilitate requirements specification. This is similar to the work presented in [10], in which the authors propose a metamodel for requirement traceability and establishes a link between requirement model, solution model and V&V model flows and affords full traceability of requirements, including those set for heterogeneous models. The focus of these two works is most on traceability matrixes between models, and not on MARTE stereotypes for requirements specification.

Model-based approaches are often applied to contribute to handle complexity of RTS development. Research presented in this article is about requirements development, mainly the activities of documenting and analyzing user requirements for software systems. In this article, the proposal is to combine UML, SysML Requirements diagram and MARTE stereotypes to improve activities of requirements specification for RTS. Among the objectives of this research is the representation, in a single metamodel, of an appropriate model to create requirements specifications for RTS with additional attributes included in the SysML Requirements diagram. The MARTE profile provides key resources to specify non-functional requirements for RTS, as for instance, temporal aspects and constraints. The SysML profile alone does not present the representation of temporal, behavioral and performance requirements. Therefore, we propose to combine SysML models with MARTE stereotypes. Thus, in addition to the factors mentioned above, we can say that the proposed approach has an important role to specify RTS at different levels of detail and levels of abstraction. The main focus is to show how these profiles can be combined and applied for modeling time, performance, system configuration, and of course, the functional and non-functional requirements of RTS. Improved tracing between requirements is proposed with new relationships between requirements. A road traffic control system is used as case study to demonstrate the applicability of the proposed approach.

2 Theoretical background on MARTE and SysML

Among the proposed UML profiles, two are used in this article. SysML, because of its Requirements diagram, and MARTE, due to its several stereotypes used to specify non-functional properties. Both profiles are briefly described in this section.

2.1 MARTE

Architecture of the MARTE *profile*, depicted in Fig. 1, consists of three main packages. **MARTE Foundations** aims at defining fundamental concepts for embedded RTS, including concepts that are useful as a general basis for the description of most of the elements. **MARTE Analysis Model** provides concepts for model verification and validation [22]. **MARTE Design Model** provides support to conduct a detailed specification of a RTS.

In addition to these packages, the MARTE profile proposes numerous other subpackages. **MARTE Foundations** relates to the scope of this work (more specifically the first three subpackages) and, for this reason, these are briefly described as follows.

– **Time Modeling—Time**: allows modeling of time and related structures, including concepts related to logical

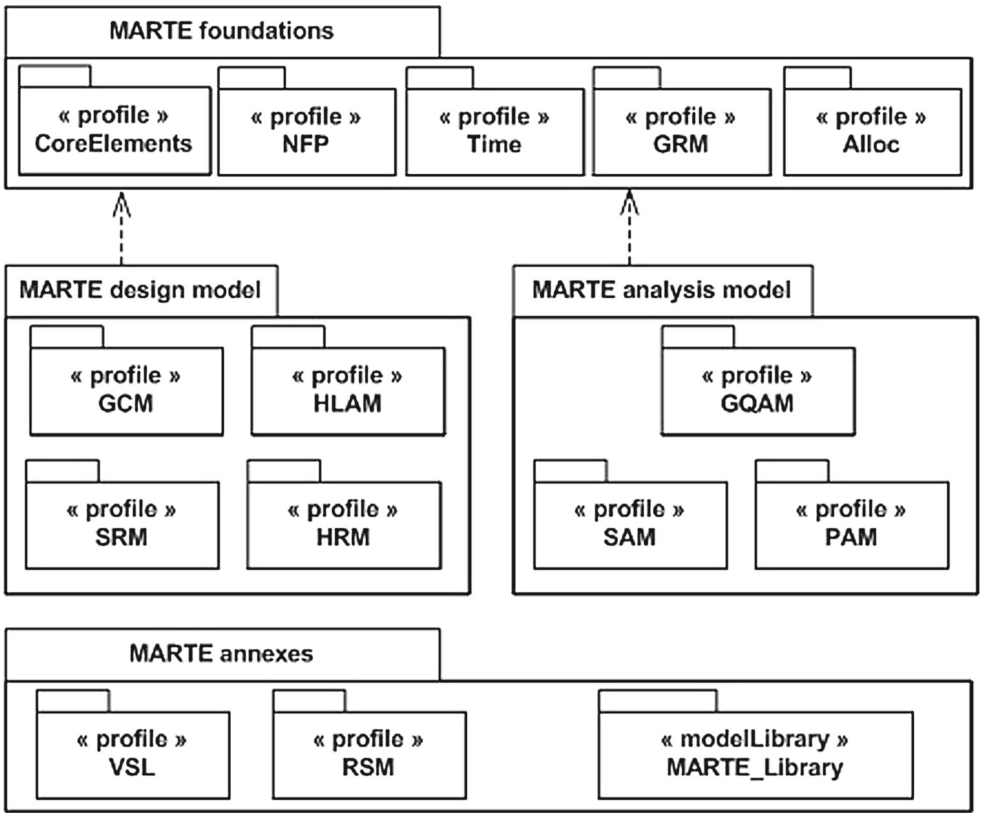


Fig. 1 MARTE profile [29]

time, physical time, representation of instants representing time bases and occurrences of events over time.

- **Non-Functional Properties Modeling—NFPs**: offers paths to specify non-functional properties of RTS, such as memory usage and power consumption.
- **Core Elements**: provides basic elements for behavioral modeling and semantic representation of its running time.
- **Allocation Modeling—Alloc**: an application element in MARTE can be a service, computation or a function of the operating system. An execution platform is a collection of connected resources representing the hardware architecture. It consists of hardware resources such as memory resources, communication devices and computing resources.
- **Generic Resource Modeling—GRM**: provides stereotypes and tagged values to represent features such as communication means, computing resources and storage resources. It also includes resources that are needed to deal with the modeling of execution platforms at different levels of abstraction and modeling.

2.2 SysML

Although UML lacks important modeling characteristics for systems and is considered to be more adequate to modeling software elements, it appeared as a good candidate to be extended with elements that could meet the most important requirements and characteristics for modeling systems that involves not only software, but also elements such as mechanical components, hardware and electronic parts.

SysML is a UML profile [30] applicable to various types of engineering applications, enabling specification, analysis, design, verification and validation of complex systems. Results from article [41] describes SysML as a suitable modeling language for systems architecture. SysML profile is supported by OMG (*Object Management Group*) and by INCOSE (*International Council on Systems Engineering*) [30].

SysML Requirements diagrams provide support for modeling individual requirements and their relationships. The basic graphical node to design requirements diagrams is depicted in Fig. 2. SysML Requirements diagram allows to model requirements relationships in various manners. These relationships are briefly described as follows.

A requirement that is contained within another requirement is described by the **Hierarchy Requirement Relationship**. This element means the relationship allows relating requirements in different hierarchical levels. A common example is the hierarchical gradual specification of requirements, from high-level business requirements into more detailed software requirements. This relationship is represented by a circle with a plus sign inside (⊕).

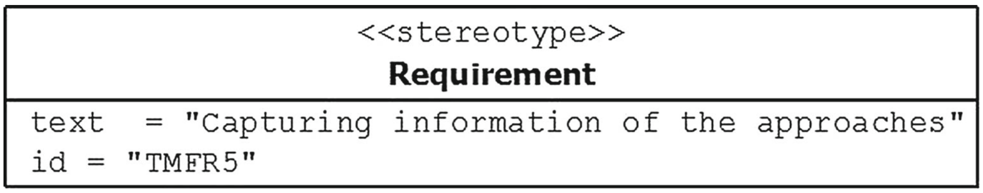


Fig. 2 Basic node for SysML Requirements diagrams

Each design element of the model has as purpose, directly or indirectly, to satisfy a requirement of the system. The **Satisfy Relationship** describes these situations, by modeling design elements that performs/satisfies a specific requirement.

Copy Relationship, denoted as a dotted arrow pointing from the copy to the original and the stereotype ≪*copy*≫, describes a requirement that is a copy of another requirement. This relationship is useful for reusing a particular requirement in another context.

Derive Requirement Relationship, represented by stereotype ≪*deriveReqt*≫, describes a requirement that was derived from another requirement. This relationship explicitly shows when a requirement can result in other requirements.

Verify Relationship, denoted with a dashed arrow, pointing from the test case in the direction of a requirement, with stereotype ≪*verify*≫, connects a test case with the requirement that is verified by this test case. A test model usually defines a large number of test cases that test whether requirements are properly implemented in the system.

Refine Relationship specifies that one model element describes properties of a requirement into more detail. For example, a functional requirement can be refined by one or more use cases.

Trace Relationship is a relationship between a requirement and an arbitrary model element. It describes a general relationship for traceability reasons.

3 Combining SysML and MARTE in a metamodel

The metamodel proposed in [37] is extended in this article, as presented in Fig. 3. SysML Requirements diagram has been extended with additional relationships to allow a new representation for software requirements. New attributes for extended requirements take into consideration specifications of the IEEE 830-1998 standard for documenting software requirements [19].

An extended requirement, represented by stereotype ≪*ExtRequirement*≫ is proposed here. In addition, derived from this extended requirement, an extended requirement for non-functional requirements is proposed, represented by ≪*ExtRequirementNFR*≫, with additional attributes. Three types of non-functional requirements are proposed in the metamodel, as depicted in Fig. 3.

Original attributes of SysML Requirements are ID (title) and text. Title is unique and briefly indicates the requirement context, and the text attribute is a short explanation of the requirement. New defined attributes for ExtRequirement are classification, type, priority, abstractLevel, constraint, scenario, creationDate, modificationDate, versionNumber and stakeholder. They are all optional in a model.

Attribute **classification** describes whether the requirement is functional or non-functional.

Attribute **type** indicates special features of a requirement, as for instance, if it represents a special state, if it relates to events, or if it represents timed elements (clocks). In order to model SysML Requirements diagram with MARTE stereotypes, it is mandatory to import packages from MARTE Foundations that relate to the behavior of an element of the domain (CoreElements package), including non-functional properties (NFPs) and elements that relate to the time structure (package Time).

Attribute **priority** defines the relevance of a requirement in relation to the other, i.e., indicating the order in which requirements should be addressed.

Attribute **constraint**, of type boolean, shows requirements that have some type of restriction. If it is set to *true*, the identifier (ID) and the detailed description of this restriction are contained in a table of restrictions.

Attribute **level** indicates the classification level of each requirement in the hierarchy.

Scenario is an attribute of type String which basically identifies the scenario to which the requirement is related.

Attributes **creationDate** and **modificationDate** are of type string with the purpose to annotate creation date of the requirement and date on which it was modified.

Attribute **versionNumber** is useful to keep track of multiple versions of the requirement. This is important when a stakeholder changes the requirement. In this case, everyone knows how many times the requirement has been changed.

Attribute **stakeholder**, of type string, relates directly to attributes modificationDate and versionNumber since it defines the responsible for every possible change requested for a requirement, on a given date, including its creation.

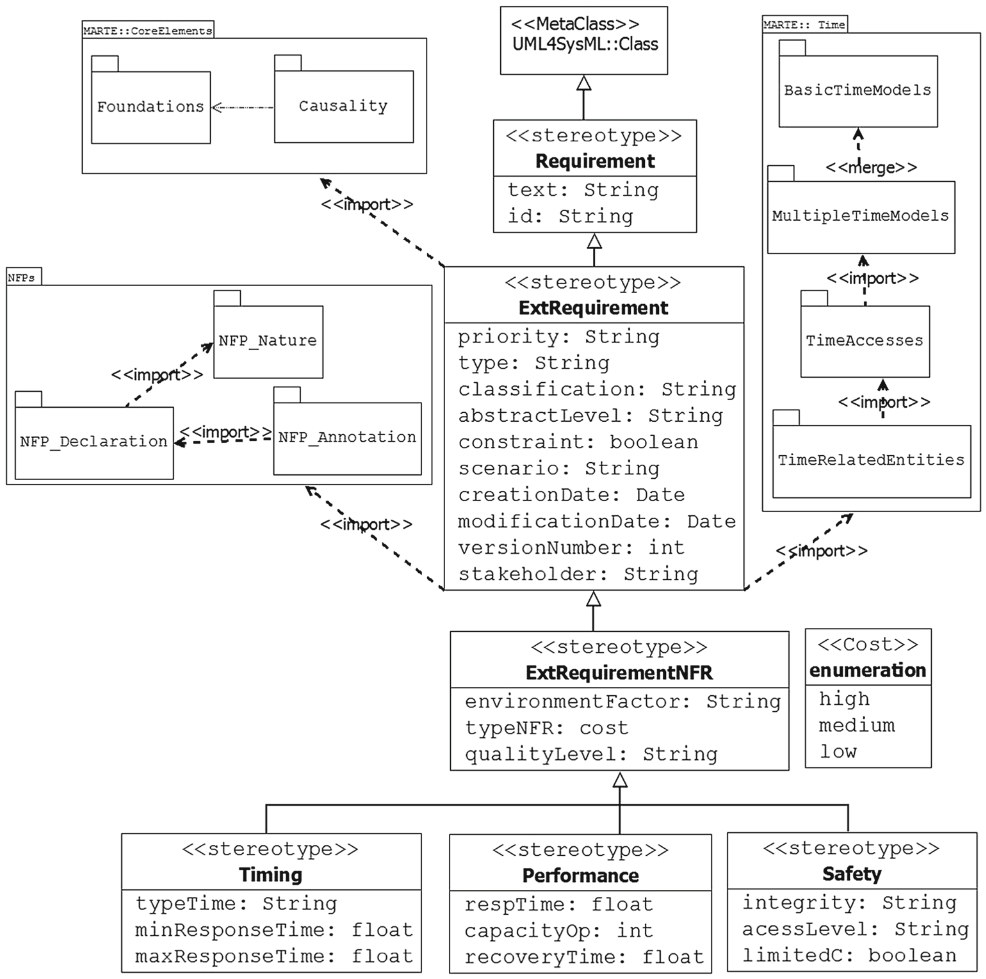


Fig. 3 Metamodel for SysML and MARTE

Stereotype ExtRequirementNFR is used to describe non-functional requirements. Proposed attributes are environmentFactor, cost and levelQoS.

Attribute **environmentFactor** determines whether a requirement is dependent on an external factor to be implemented. In general, this attribute allows anticipate external features to the system, in development, and that has an impact on the development/design of it. It is an attribute of type string and brings a brief description of the dependency factor. Attribute **cost** is an enumeration type that defines Strings used for specifying the possible values for the development of a system requirement. The enumerations are the type high, medium or low and allows for the establishment of criteria of costs to satisfy a requirement that influences directly in those decisions concerning the viability of its development. Attribute **qualitylevel** indicates the level of quality required for the requirement, and cal also assume values high, medium or low. With this semantics, for the attribute **qualitylevel**, it becomes possible to predict whether standardization strategies should be taken/established during system development activities.

Performance type has three attributes. Attribute **respTime** indicates the maximum response time associated with a requirement. Its value allows for establishment of which performance level is to be associated or is to be guaranteed by the requirement. Attribute **capacityOp** indicates the number of simultaneous operations that are allowed on a given time period (e.g., number of reports generated for storage, operations per second). Attribute **recoveryTime** describes the maximum time required for recovery from a failure.

Timing type of non-functional requirement relates to the description of time in software. Its attributes are **typeTime**, which can assume values such as physical time or logical time, **minResponseTime** and **maxResponseTime**, which are used to describe timing constraints of a requirement.

Safety type of non-functional requirement has attributes **integrity** (level of integrity that must be guaranteed), **accessLevel** (establish the level of access of stakeholders to a function) and **limitedC**, which enables demonstration of whether communication should be limited between this requirement and other functions/modules of the system.

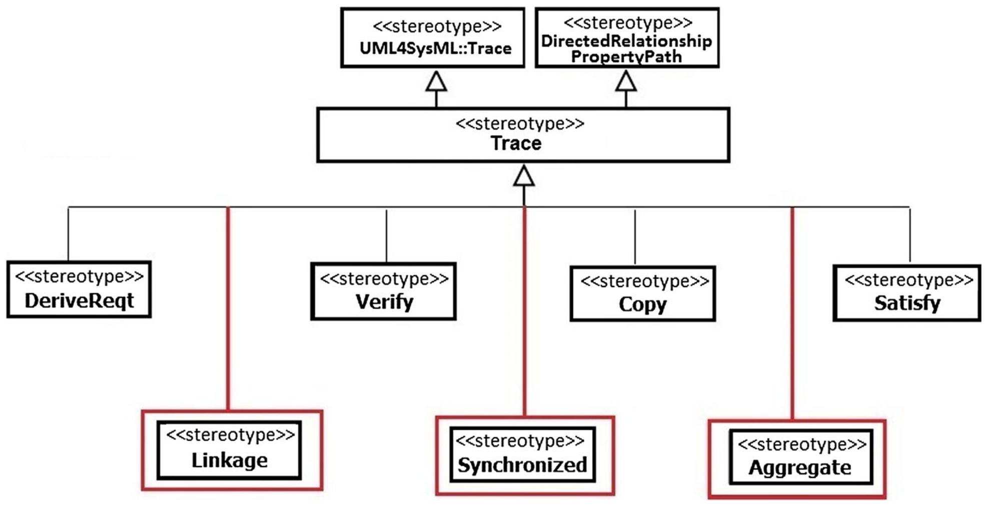

<<stereotype>> UML4SysML::Trace
<<stereotype>> DirectedRelationship PropertyPath
<<stereotype>> Trace
<<stereotype>> DeriveReqt
<<stereotype>> Verify
<<stereotype>> Copy
<<stereotype>> Satisfy
<<stereotype>> Linkage
<<stereotype>> Synchronized
<<stereotype>> Aggregate

Fig. 4 Metamodel of extended relationships

3.1 Extended relationships to the SysML Requirements diagram

SysML Requirements diagram allows several ways to represent requirements relationships, as previously described. In this work, three new stereotypes are proposed. The extended model and the new developed relationships, from the extensions to the basic relationships of SysML, are able to represent requirements at different levels of abstraction, correlated requirements at the same level and, also, synchronism between requirements to be implemented as depicted in Fig. 4.

Stereotype ≪*linkage*≫ (Fig. 5), represented by an arrow and a dotted circle with an internal dotted cross on both ends (graphical notation this becomes necessary for differentiating the new established relationship to the UML nesting relationship), has the purpose of improving the activity of tracing requirements. The linkage relationship explicitly shows, in the created models, the interconnections of a requirement at any level of the requirements hierarchy. Thus, it is possible, for example, to determine trace of a requirement in high level toward their functional and/or non-functional requirements.

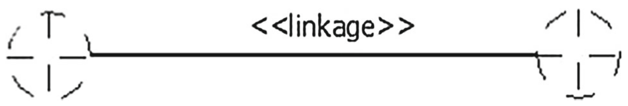


Fig. 5 Relationship linkage

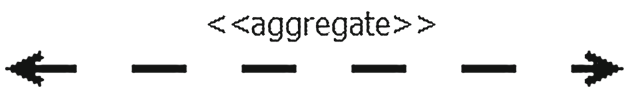


Fig. 6 Relationship ≪*Aggregate*≫

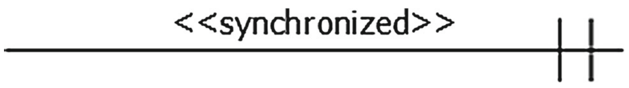


Fig. 7 Relationship ≪*Synchronized*≫

Stereotype ≪*aggregate*≫ (Fig. 6) is proposed in order to explicitly demonstrate correlated requirements (in the same classification level) with a requirement one level up; i.e., requirements that are represented with this stereotype will together provide functionality expected by a higher requirement and are strongly bonded together. Therefore, requirements connected through the ≪*aggregate*≫ relationship are strongly integrated and supply, together, a functionality proposal by a higher-level requirement. This relationship allows to represent the decomposition of requirements and/or its grouping to compose subsystems. Thus, the ≪*aggregate*≫ relationship allows, for analysis and study, the modeling of functionalities of a system in an aggregated manner.

Finally, new stereotype ≪*synchronized*≫ is designed to be used by non-functional requirements that need to describe in addition to performance constraints, the ability to be processed/executed concurrently. Its graphical representation is depicted in Fig. 7.

3.2 MARTE stereotypes

In most cases, concepts defined in the domain for subpackage *CoreElements*, subpackage *Non-Functional Properties* and subpackage *Time* presented in the last section are represented in MARTE through a stereotype that extends a UML modeling element. Thus, the UML extensions required for supporting the concepts defined in MARTE and the stereotypes are described in this subsection.

Table 1 Package MARTE *Foundations*: stereotypes of *CoreElements*

Package	Domain model	Stereotype	Definition
Causality	Modal behavior	Configuration	Represents the system configuration, can be defined by a set of elements, system as sets and/or a set of operating parameters
Causality	Modal behavior	Mode	Identifies an operating segment within the runtime system that is characterized by a determined configuration
Causality	Modal behavior	ModeBehavior	Specifies a set of mutually exclusive modes. Its dynamics is represented by connection modes by means of *Mode Transitions*
Causality	Modal behavior	ModeTransition	Describes the modeled system in switching mode. *ModeTransition* may be produced in response to a *Trigger*

Table 2 Package MARTE *Foundations*: stereotypes of *non-functional properties*

Domain model	Stereotype	Definition
NFP_Anotation	Nfp_Constraint	Aims to apply a condition or restriction to modeled elements. Restrictions for NFP support textual expressions to specify assertions about performance and other characteristics of embedded systems
NFP_Declaration	Nfp	Intended to declare, qualify and assign data types extended to NPFs values
NFP_Declaration	NfpType	A NfpType is a type whose instances are identified only by specifications of NFPs values
NFP_Nature	Dimension	Establish a relationship between a quantity and a set of base quantities in a given system quantity
NFP_Nature	Unit	A qualifier of values measured in terms of magnitudes of other quantities

The set of extensions used to support *Core Elements* (Table 1), *NFPs* (Table 2) and *Time* (Table 3) for UML modeling is organized and addressed according to the application context of domain concepts. In Sect. 4, only elements relating to the requirements specification of RTS are discussed.

4 A case study on road traffic systems control

In this section, a subset of a list of user requirements for a road traffic management system (RTMS) is first presented using natural language and is further modeled with the proposed approach.

4.1 Background on road traffic control systems

Current control equipment for traffic signals can provide a wide variety of resource capabilities usually organized in fixed time, actuated time and adaptive time [23,39].

Within **Fixed-Time Signals**, values of the cycle time, duration and sequence of phases have fixed calculated values based on historical intersection data flow. According to [39], traffic signals that use fixed time are more affordable to purchase, install and maintain than actuated time. However, this type of control does not allow a proper reaction to the fluctuation of traffic during the day.

Actuated Traffic Signals vary the green phase based on traffic demands measured by road sensors.

For an actuated control, there are three time parameters: the minimum time for the green phase, the extension of green phase and the maximum time for the green phase. Independent of demand, the green phase is defined for the minimum time. Depending on the flow of vehicles, and if the maximum duration of the green phase is not achieved, then the green phase time can be extended.

This example focuses on the requirements needed to provide control capability for actuated intersections with interconnection of traffic signals, i.e., the operation of traffic signals in a network. The set of requirements for the design of a coordinated traffic control system is presented in Sect. 4.2.

4.2 Systems requirements

A subset of a list of requirements for a road traffic management system (RTMS) is used as input document in this case study. Requirements are written exclusively using natural language to be further modeled and analyzed. The list of requirements presented in Table 4 is a subset from a document which contains 137 atomic requirements for a RTMS. This subset of requirements focuses on capabilities of an actuated controller and, also, the functional and non-functional requirements involved with synchronization of actuated controllers in a road network.

Table 3 Package MARTE *Foundations*: stereotypes of *time*

Domain model	Stereotype	Definition
TimedRelatedEntities	TimedElement	Abstract stereotype used for associating/referencing one(s) Clock(s) to one model element
TimeAccess	Clock	Introduces a general concept of clock and represents an instance of clockType which provides access to time
–	ClockType	A classifier for Clock related with time. Defines attributes to specify the nature of time (discrete/dense) or system of time (physical/logical)
TimeAccess	TimedValueSpecification	Specification of a set of instances of *Time Value*. Property interpretation can force the interpretation of this value as a duration or specification of instants. Like a *TimedElement* one *TimedValue Specification* makes references to Clock
TimedRelatedEntities	TimedConstraint	Represents constraints imposed at any value instant or in duration value associated with model elements linked to clocks
TimedRelatedEntities	ClockConstraint	The objective is to impose dependencies between Clocks or among types of Clocks. Like a *TimedElement* one *ClockConstraint* makes references to Clock
TimedRelatedEntities::-TimedEventsModels	TimedEvent	Represents events whose occurrences are explicitly related to Clocks. Repetition refers to a repetition factor, i.e., the number of successive occurrences from *TimedElement*
TimedRelatedEntities::-TimedProcessingModels	TimedProcessing	Represents activities where there is knowledge of the start time and end or with a known duration whose instants are explicitly linked to clock
TimedRelatedEntities::-TimedObservation	TimedInstantObservation	Denotes an instant in time associated with occurrence of an event and observed for a given clock
TimedRelatedEntities::-TimedObservation	TimedDurationObservation	Denotes some time interval associated with execution, request or occurrence events observed in one or two clocks. For being specialization of *TimedElement* the *TimedDurationObservation* makes references to Clock
–	TimedDomain	Refers to model elements that can refer to clocks to express that their behavior depends on time

4.3 Proposed scenarios

SysML Use Case diagram describes the usage of a system by its actors in order to achieve a goal. Figure 8 presents major functions for the proposed system and allows for the representation of external entities exercising influence in a scenario (which describes a set of requirements).

It is possible to represent the interconnection of a Use Case with a SysML Requirements diagram by using the relationship refine (Fig. 9). This relationship provides the capability for reducing ambiguity in requirements since it can show the relationship of a requirement with scenarios.

Figure 10 depicts an example in which a Use Case proposed in Fig. 8 is refined to a set of requirements described in the SysML Requirements diagram. In this case, the scenario *Synchronized Semaphore* is interconnected, through the refine relationship, with requirements *greenSych*, *synchronizationState*, *miniBlockage* and *greenSychMM*.

Table 5 describes a proposal for tracing all scenarios of Fig. 8. Table 5 is prepared based on the analysis of natural language specified in Table 4, through the description of the use cases performed in Fig. 8 and also from the drawdown of stakeholders that interact with each scenario of the system. After this analysis it was possible to conceive manually the tracking all use cases that relate to the overall requirements of the system.

Classification proposed in Table 5 improves the representativeness of the SysML Refine relationship (in this table, TS stands for traffic signals, OP for operator, EV for emergency vehicle, MA for maintainer and S for sensor). Thus, the relationships between requirements are documented since establishment of initial phases of the system development and can be validated and traced along the development process.

5 Modeling requirements for a road traffic control system

Figure 11 depicts how to apply the SysML Requirements diagram with MARTE stereotypes for modeling non-functional properties, runtime semantics that are suitable for real-time and embedded systems, more specifically to the proposed requirements for a traffic control system as specified in Table 4.

Main requirements and the respective modeling using the SysML Requirements diagram extended with MARTE stereotypes are presented in Fig. 11.

Table 4 A subset of requirements for a road traffic control system

ID	Requirement name
TM1	The system must control the standard of vehicular traffic at the intersection
TM2	The system must allow synchronization of traffic signals
TM3	The system should collect all kinds of information of the road approaches in order to properly evaluate these data
TM4	The system must allow management of traffic history
TM5	The system must actuate in response to intersection traffic flow
TM6	The system must coordinate green time of intersections in a synchronous way
TM7	The system must have the emergency preemption mode, i.e., preferred movement of emergency vehicles
TM8	The system must allow the control of the intersection in response to manual commands
TM9	The system must allow control of intersection in response to replace remote commands
TM10	The system must control the semaphore of the intersections
TM11	The system of the intersection should be able to interact with the software control panel
TM12	The system must calculate the delay to the controllers of intersections
TM13	The system must optimize the flow of traffic
TM14	The system must set the mode/state flag in response to the current processing of intersection control
TM15	The system must minimize blockages along the highway
TM16	The system must efficiently change plans for liberation of platoons
TM17	The system must check traffic demand with maximum precision
TM18	The system must allow for incident management
TM19	The system must configure green phase time for intersections
TM20	The system must coordinate green phase of the intersections precisely, with minimum of 50 ms and maximum of 150 ms

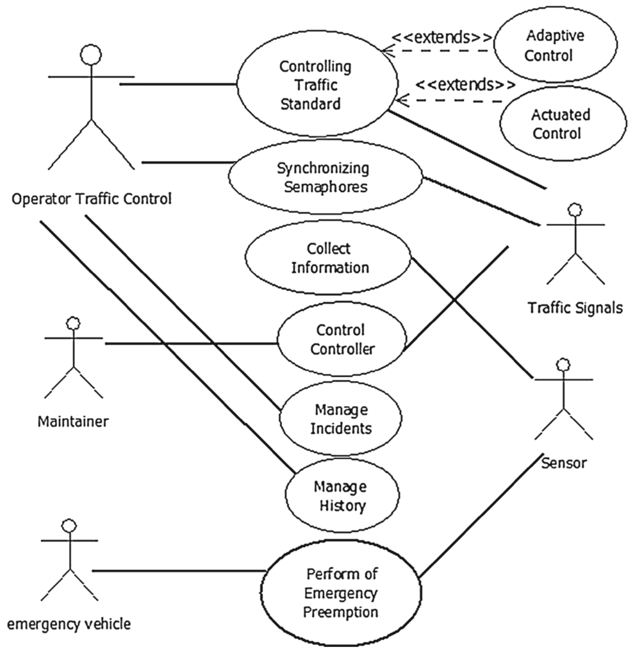


Fig. 8 SysML Use Case diagram for the traffic control system

In order to improve model understandability, two groups of requirements (Table 4) are grouped into packages in Fig. 11. Package $P1$, depicted in Fig. 12, models in detail

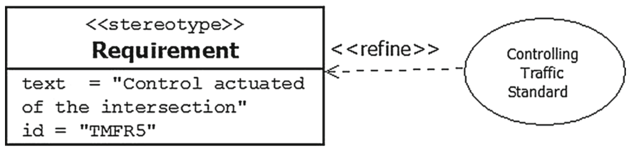

<<stereotype>>
Requirement
text = "Control actuated of the intersection"
id = "TMFR5"
<<refine>>
Controlling Traffic Standard

Fig. 9 Refine relationship

requirements $TM3$, $TM4$, $TM7$, $TM8$, $TM9$, $TM11$, $TM13$, $TM15$ and $TM16$ and represents scenario which has as purpose to define basic properties to traffic control. Package $P2$, depicted in Fig. 13), is composed of requirements $TM12$, $T19$ and $TM20$ which, when grouped, establishes the traffic signal phases, its synchronization and performance criteria.

Requirement *TM1* has as type stereotype *Mode* (of *Causality::ModalBehavior*), indicating the necessary logic to represent and to control an operating segment as, for instance, a traffic control system for all active entities/elements of the operational fragment. This requirement has a SysML relationship ≪*hierarchy*≫ with the basic requirements for a traffic control system, including requirements *TM3*, *TM4*, *TM5* *TM7*, *TM8*, *TM9*, *TM11*, *TM15* and *TM17*.

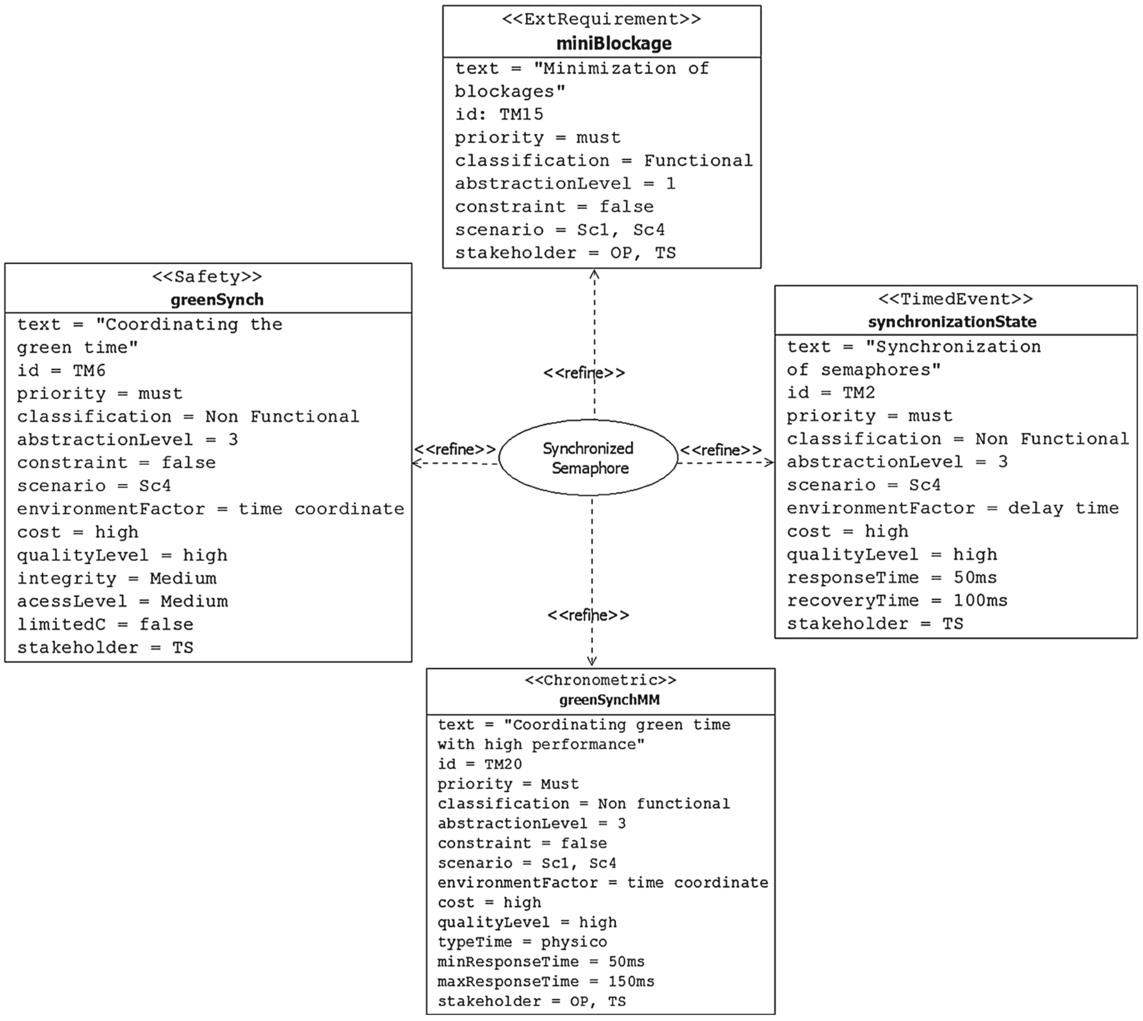


Fig. 10 Synchronizing semaphores and its requirements

Table 5 Tracing scenarios

Scenario name	ID	Actor related	Related requirement
Controlling standard	Sc1	OP	TM1, TM4, TM10, TM11, TM12, TM13,TM14, TM15, TM16, TM19, TM20
Adaptive control	Sc2	TS	–
Actuated control	Sc3	TS	TM5
Synchronizing semaphores	Sc4	TS	TM2, TM6, TM15, TM20
Collect information	Sc5	OP, S	TM3
Control controller	Sc6	MA, TS	TM8, TM9, TM11
Manage incidents	Sc7	OP	TM18
Manage history	Sc8	OP	TM17
Emergency preemption	Sc9	EV, TS	TM7

Requirement *TM7*, of type TimedEvent (of *TimedRelatedEntities:: TimedEventsModels*), relates to an event whose start and end are not defined a priori. However, the decision to attend this occurrence is directly linked to a Clock. The *modePreemp* requirement (*TM7*) has a constraint represented by the ≪*nfp_Constraint*≫ stereotype (from *NFPs::*

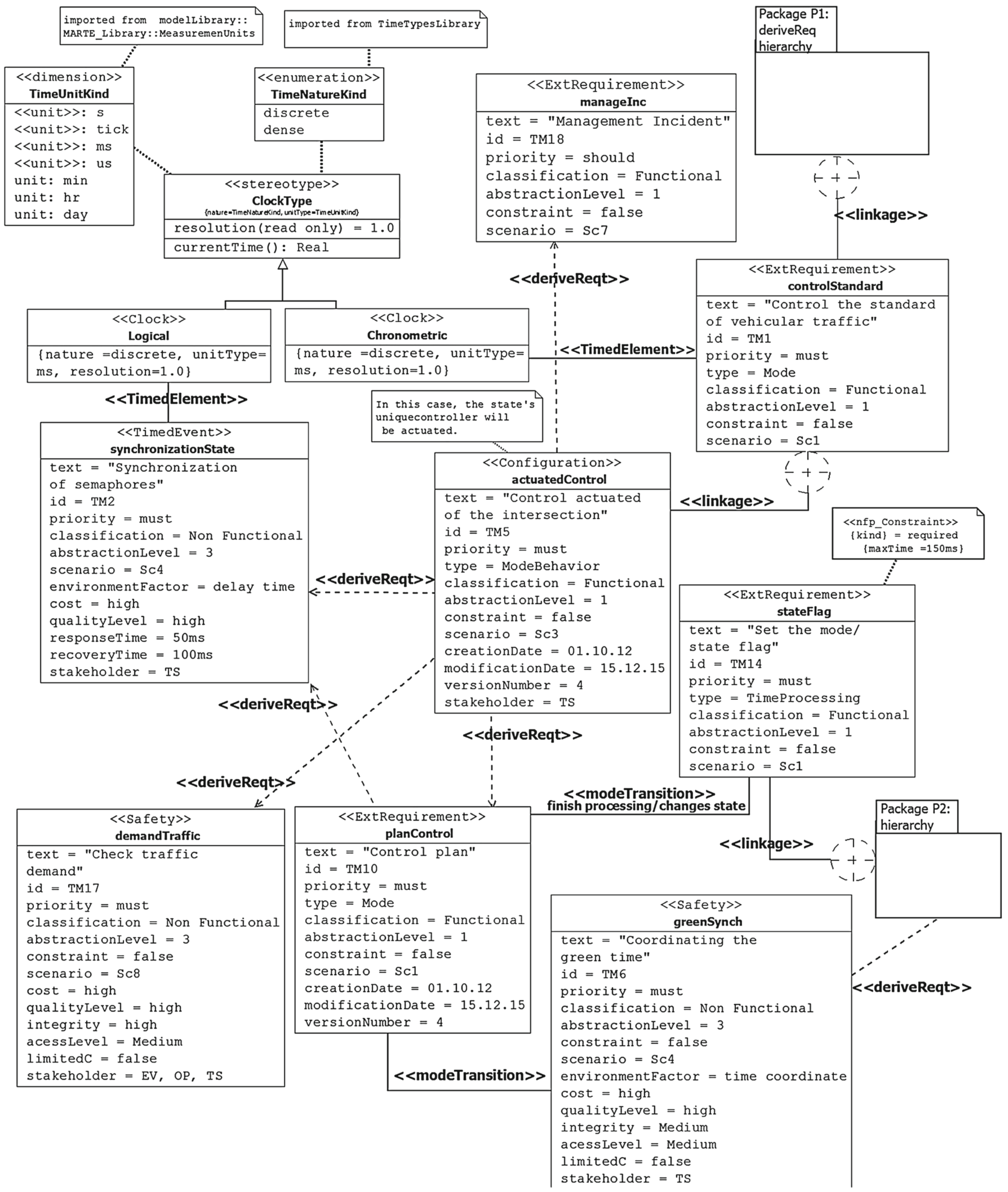


Fig. 11 Complete model

Nfp_Constraint) which applies a temporal restriction to "ensure performance" of a critical requirement. For instance, the priority passage of emergency vehicles is indicated by attribute *kind* = *offered*, which indicates the value space to support/restrict this requirement.

Requirement *TM1* is related to requirement *Chronometric* using stereotype ≪*TimedElement*≫ (of *Time::TimedRelatedEntities*) which is a specialization of *ClockType*, both belonging to package *Time::TimeAcess*. Stereotype ≪*TimedElement*≫ is important in this context, as it should be used whenever it is necessary to associate a (few) Clock(s) to a model element (in this case the *TM1* requirement). Requirement *TM1* should be elaborated within time constraints to ensure several important non-functional requirements for traffic control systems. Thus, it is necessary to create a ≪*clockChronometric*≫ stereotype enabling the *TM1* access to time structure. Attributes of this clock are *nature*, an enumeration of *TimeNatureKind::TimeTypesLibrary*, which

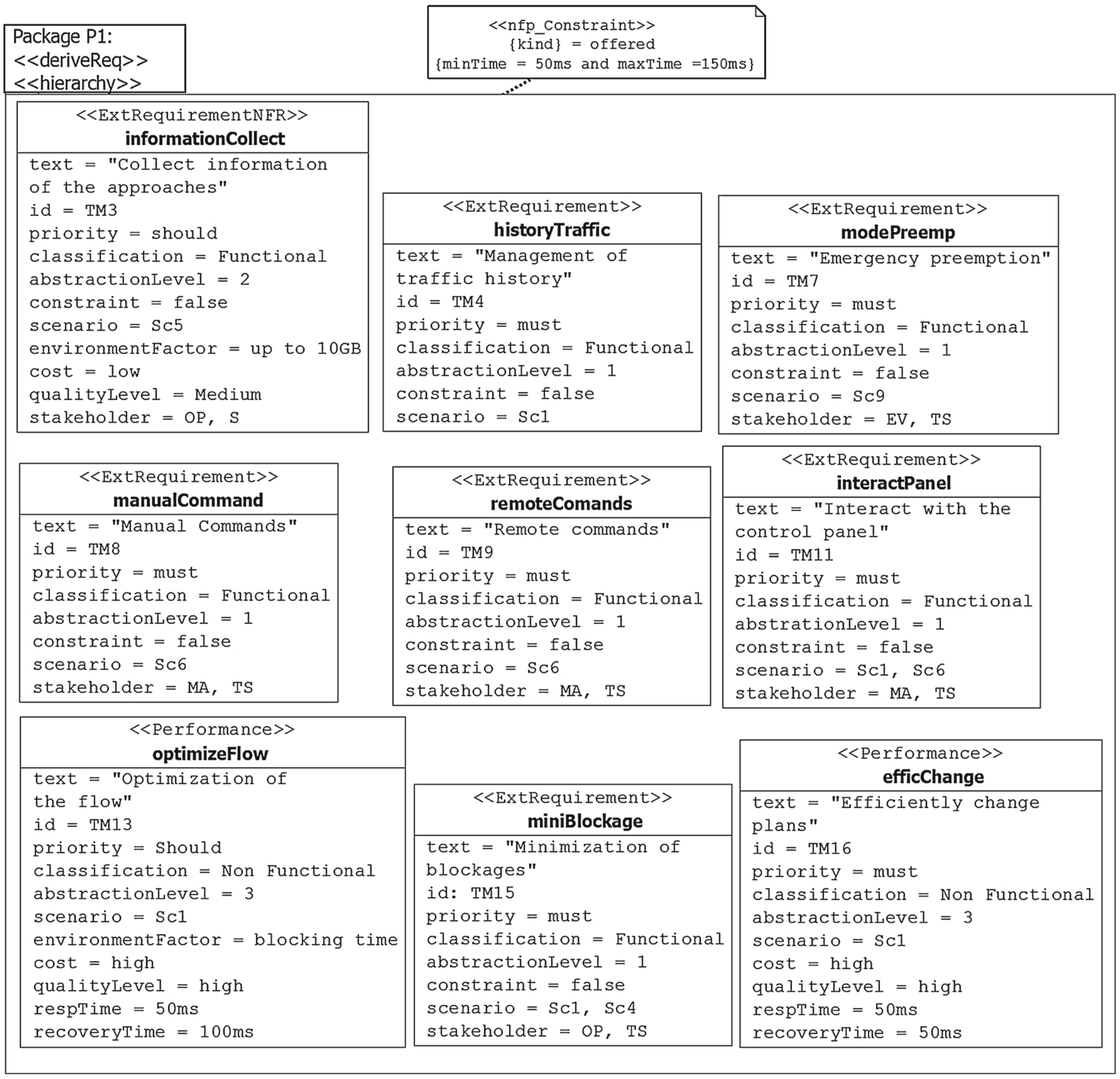


Fig. 12 *P*1 package in detail

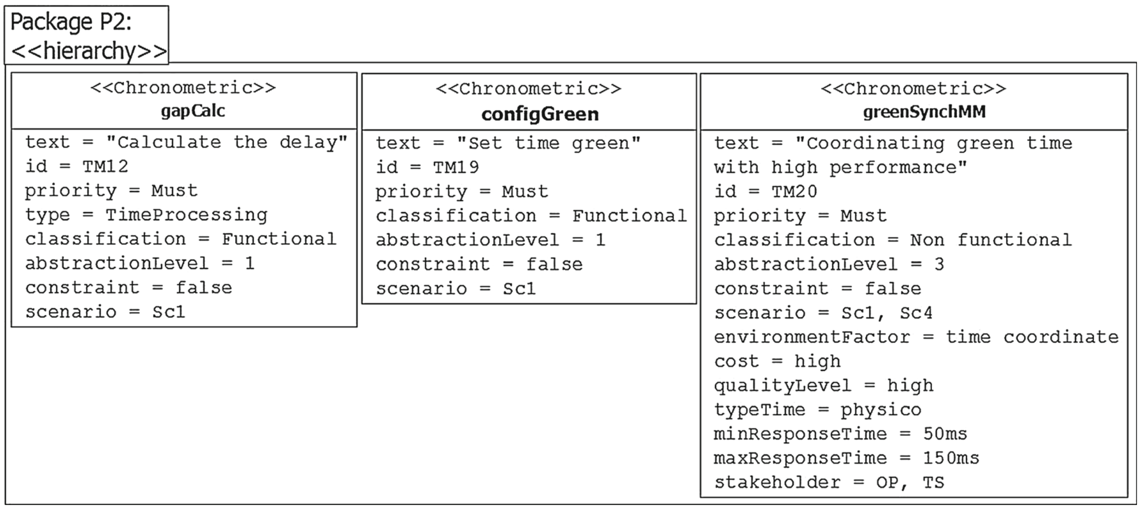


Fig. 13 *P*2 package in detail

serves to specify the discrete or dense nature of a time value (in this case time is discrete), *unitType*, a *TimeUnitKind* dimension imported from *modelLibrary::MARTE_Library:: MeasuremenUnits*, which defines the supported unit type (in this case the unit is ms), and the resolution which expresses clock granularity (in this case by default it was set to 1.0).

Control logic of the controller is of type actuated time. Requirement *TM5* relates hierarchically to *TM1* in order to define the configuration of the control system. This is represented through stereotype ≪*Configuration*≫ (*of CoreElements::Causality::ModalBehavior*). This stereotype shows the system configuration that can be defined by a set of active elements from the system. Type attribute of requirement *TM5* is *ModeBehavior* (of *CoreElements::Causality:: ModalBehavior*), because the actuated control mode is unique to the control system; i.e., it is the only one considered at the intersections of this case study.

Requirement *TM5* derives from *TM10*, which is responsible for controlling the control plans for signaling the intersections. Signaling plan changes depending on volume of road traffic. Thus, through an association with *TM14* with ≪*modeTransition*≫ stereotype (of *CoreElements::Causality ::ModalBehavior*), the signaling plan transitions to a certain state of the signals (red, green, yellow, green expanded and so on) specified at *TM14*.

Requirements *TM19* and *TM20* are stereotyped with ≪*Chronometric*≫ characterizing them as requirements that exist in a physical time. These requirements will work with physical time to configure the green time of a phase (*TM19*) and, also, coordinate this time with high performance (*TM20*) in order to ensure fluid traffic flow. Requirements *TM19* and *TM20* are linked to *TM14* (by *hierarchy* relationship of SysML).

Requirement *TM14* is stereotyped as *TimedEvent* (of *Core Elements::Time::TimedRelatedEntitiesTimedEventsModels*), since it describes events that force change and the state controller processing, whose occurrences are directly linked to a *Clock* (in this case Clock Chronometric).

Requirement *TM14*, of type *TimeProcessing* (of *Time:: TimedRelated Entities::TimedProcessing*), is important to clarify that the change of states signaled is an activity where there is great importance in knowing the initial time and finish time for each state, as well as creating strategies to ensure/get higher performance. Timing restriction for this requirement precisely demonstrates that the same processing must have a minimum quantitative level (specified with *kind* = *required*) to run with at most 150ms.

Requirement *TM12*, of type *TimeProcessing* (because its processing time instant refers explicitly to clocks), refers to a structure of physical time. It is through this requirement that the delay (delay/offset) is acknowledged between a controller of an intersection and the subsequent intersections.

Requirement *TM2* was stereotyped as a TimedEvent by presenting a situation or criterion of the controller that should trigger/capture events that synchronize the traffic plans of different interconnections actuated form (note TM5).

MARTE enables both logical time and physical clocks modeling. For this reason, a structure of time that specializes ≪*clockType*≫ and sets the logical time necessary to demonstrate that the *TM2* requirement relates to synchronizing semaphores is created at an intersection network. It does not carry a long presigned physical time; it can only be seen as reading and detecting the flow of the processing approaches, the setting of the control plan strategy and so on. Therefore, it is not explicitly a physical time, and processing logic depends on several other elements.

6 Extending a tool for requirements modeling with SysML

ArgoUML (available in [3]) is a software tool used for activities of analysis and design of Object-Oriented software systems using UML. ArgoUML is an open source software, multiplatform, based on GTK+ (one multiplatform tool kit for creating graphical user interfaces) for creating diagrams and released under the license BSB (Berkeley Software Distribution). It was developed using the Java language and is composed of approximately 2.000 class and 15.000 methods [3].

To facilitate the analysis and understanding of code, enabling the modification and adaptation of the tool for modeling proposed in this article, Software Reconnaissance techniques [49] were applied, in addition to analysis of the documentation of ArgoUML, including available tutorials and examination of the source code. Techniques of Software Reconnaissance were used to analyze execution traces to find components of interest [38]. This technique makes it easy to find in the program the locations in which a particular function of interest is implemented.

In order to implement new functions in the ArgoUML tool, the technique code clone [9] was applied. Code clones are a portion of the source code reused in another part of a program. A SysML Requirements diagram is an extension of the UML Class diagram. In this context, the adaptation of the source code of ArgoUML was performed, in particular the class diagram (package *UMLClassDiagram* of the ArgoUML), to make possible modeling requirements of SysML diagram.

Changes in ArgoUML tool, highlighted in Fig. 14, were performed by adding a new button to the toolbar. This button generates a command that is sent to a controller (CoreFactoryMDRImpl) which effectively creates the SysML Requirements diagram on ArgoUML.

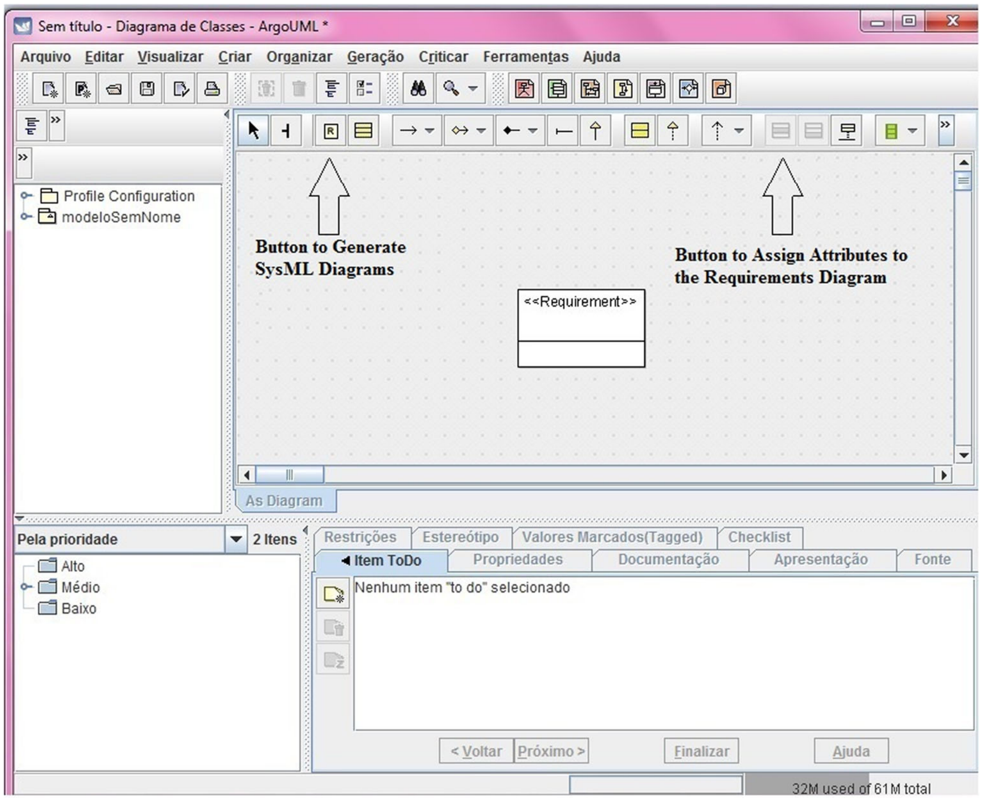

Sem título - Diagrama de Classes - ArgoUML *
Arquivo Editar Visualizar Criar Organizar Geração Criticar Ferramentas Ajuda
Profile Configuration
modeloSemNome
Button to Generate SysML Diagrams
Button to Assign Attributes to the Requirements Diagram
<<Requirement>>
As Diagram
Pela prioridade
2 Itens
Alto
Médio
Baixo
Restrições Estereótipo Valores Marcados(Tagged) Checklist
Item ToDo Propriedades Documentação Apresentação Fonte
Nenhum item "to do" selecionado
< Voltar Próximo > Finalizar Ajuda
32M used of 61M total

Fig. 14 Interface of extended ArgoUML tool

7 Comparison with related work

After an extensive literature review about the use of *profiles* SysML, MARTE, and *profiles* SysML and MARTE together, 21 works regarding requirements engineering were found from 2007 to 2014. These works are listed below: 1. [44], 2. [25], 3. [45], 4. [7], 5. [47], 6. [43], 7. [24], 8. [8], 9. [13], 10. [34] , 11. [6], 12. [51], 13. [11], 14. [21], 15. [42], 16. [2], 17. [28], 18. [14], 19. [36], 20. [16] and 21. [26].

The 21 papers previously listed were selected studies that addressed modeling strategies similar to the proposed in this work, that is, works describing the application of a methodology using profiles SysML and MARTE. Works that are most easily compared with the strategy proposed in this article are, respectively, [2,14,16,26,28,36].

A set of comparison criteria was designed to evaluate the approaches already described in the literature review. Comparison criteria, mentioned above, were chosen based on the specifications of the standard [19] for modeling software requirements. Comparison of works listed previously to the approach proposed in this article is shown in Table 6. Symbols used in Table 6 are as follows:

■—means the evaluated criterion is fully satisfied;
⊡—means the evaluated criterion is partially satisfied;
☐—means the criterion evaluated is not satisfied.

Table 6 presents strong points in favor of the approach proposed in this work that utilizes profiles SysML and MARTE in conjunction for describing requirements of RTS.

The nearest researches which can be compared with this article are presented in [14,26]. However, the first article presents a broad theoretical discussion of the use and adequacy of profiles MARTE and SysML to deal with multidisciplinary aspects in the field of embedded systems. It is worth mentioning that the fields, in Table 6, labeled with "*" are justified by the fact that despite being marked as partially satisfied criteria this research relates to a theoretical study about the use and applicability of these profiles. Our proposal specifically made suggestions regarding the applicability of SysML and MARTE in a single framework.

In [26], one can observe the application of SysML and MARTE in the context of the elicitation, analysis and specification, validation and management requirements in the approach named MDEReq. In this case, designers have an effective control of requirement changes and its impacts on other requirements or other design artifacts. In the proposed approach, traceability matrixes and SysML Requirements diagram are created for managing requirements process. In this research, the SysML Requirements diagram was used as designed in SysML (no extension or detail). Time and GRM packages of MARTE Foundations were applied to design

Table 6 Comparison of approaches for modeling requirements

Criteria of comparison	Paper ID 16	Paper ID 17	Paper ID 18	Paper ID 19	Paper ID 20	Paper ID 21	Our approach
Grouping of requirements	□	□	□	□	□	□	■
Classification of requirements	■	□	□	⊡	⊡	⊡	■
Specific requirement diagram	■	□	□	⊡	□	□	■
Extensibility of template	□	□	■ *	□	■	■	■
CASE tools available	■	⊡	□	■	⊡	⊡	⊡
Identification of conflicts	□	□	⊡ *	⊡	⊡	■	□
Functional requirements modeling	⊡	⊡	⊡ *	■	⊡	⊡	■
Non-functional requirements modeling	⊡	⊡	⊡ *	⊡	■	⊡	⊡
Graphic modeling	■	■	□	■	■	■	■
Time modeling	⊡	⊡	⊡ *	⊡	⊡	⊡	■
Traceability with design	■	■	□	■	⊡	■	■
Traceability between requirements	⊡	□	□	□	□	■	■
Relationships between requirements	■	□	□	■	□	⊡	■
Relationship with the UML	■	■	■	■	■	■	■
Representation of restrictions	⊡	□	□	⊡	⊡	⊡	■
Specific methodology for requirements	⊡	□	□	⊡	□	□	■

definitions only in three of its constructors and stereotypes. Therefore, in this work, elements that define/establish events, performance, constraints and so on were overlooked.

Project MADES is described in [35,36]. This methodology was developed after current practices for developing embedded real-time systems and are applied to aviation industries and surveillance. A major contribution relates to the presentation of a complete methodology based on the combined use of SysML and MARTE for design, validation, simulation and automatic code generation while integrating aspects such as component reuse. The approach proposed in MADES [36] differs from our proposal, first by contemplating a comprehensive methodology for developing embedded systems, and second, by joining a consistent set of diagrams for specification of system requirements, initial behavior specification, functional specification, refined functional specification, specification of hardware/software, detailed specification of hardware/software and specification of allocation.

The approach proposed by [16] demonstrates the use of SysML Parametric diagram and some MARTE packages. Thus, the concepts and diagrams of SysML/MARTE complement the methodology and are not utilized in a grouped way as proposed in our article. In [16], each domain can be treated separately in different views while maintaining strong connections toward other views. MARTE profile is used to define the model of hardware architecture. In addition, the MARTE package of non-functional properties is used for setting properties such as power, voltage and frequency. SysML is used to specify, through the parametric model, the equations that define mathematical relationships between non-functional interests of different views.

8 Discussion

According to the extensive bibliographic research conducted until the present moment, there are few studies in scientific literature which focuses on the applicability of UML and SysML with MARTE stereotypes for documentation, classification and modeling of software requirements for RTS. Despite the existence of several approaches for modeling software requirements and for modeling specific requirements of real time, as described in previous sections, requirements engineering for RTS still lacks representative models that are expressive, complete and correct. As described by [14], a single profile may not be adequate to cover all required aspects as, for example, the multidisciplinary aspects in the field of RTS.

In this article, the proposed approach shows the applicability of the MARTE profile, with its stereotypes and restrictions, with the SysML profile for requirements described in greater level of detail and at different levels of abstraction, as well as classifying and tracing requirements. Thus, MARTE stereotypes specifies the notion of time in a better manner than the way presented in UML. Within the application of the proposed metamodel, in Sect. 5, it is possible to observe the contributions of this work to improve representation and documentation of requirements. The proposed metamodel is complementary to other methods of modeling software and are not intended to replace them (such as Use Cases and respective scenarios).

Among the related criteria which can be highlighted as strong points of this work are the following: **clustering of requirements** (through the new relationship ≪*aggregate*≫), the **classification of requirements** (by adding a set of key attributes to the SysML Requirements diagram), the **expressive modeling of functional requirements** and **partial modeling non-functional requirements** (through extended SysML Requirements diagram and by using MARTE *profile*), the **definition** (or redefinition from extension of SysML relationships and extension of SysML Tables), **issues of traceability** between requirements and between requirements and design and, yet, the **development of a specific methodology for requirements**, what is considered an advantage, since it specializes and defines described domain-specific criteria.

Although the proposed approach has used packages *CoreElements*, *Time* and *NFP* for modeling real-time elements, the representation of non-functional requirements and the representation of time are not covered in completeness. The reason is because even being possible to model restrictions, *clocks*, performance and safety, this work still lacks greater level of formalism for modeling non-functional requirements.

Finally, one can observe in Table 6 that the proposed approach is relevant in the context of requirements specification of RTS. Importance of modeling in industry and education has been emphasized, for instance, in [15]. In Software Engineering, using models at the design phase of a software system is becoming more frequent. However, on early stages of software development, as for instance, requirements engineering activities, modeling is still not common.

Future work will focus on using SysML Activity diagrams for modeling and simulation [27] in order to provide frames of reference for models validation [4].

9 Conclusions

As previously reported, SysML Requirements diagram shows explicitly various types of relationships between different requirements, increasing the spectrum of understanding and defining the requirements of a real-time system. However, the SysML profile by itself does not present the representation of temporal, behavioral and performance requirements, nor provides elements for explicit representation of system configurations.

MARTE profile provides key resources to specify non-functional requirements for RTS, generally time requirements. In this article, these features were made explicit by means of classification and use of MARTE stereotypes. Thus, the combination of these profiles in the presented approach indicates the complete and expressive nature of the representation of various requirements. SysML contributes with constructors to define requirements and their relationships. Besides, MARTE completes the precision of the scenario with well-formed non-functional annotations. Concepts of SysML and MARTE articulated in the SysML Requirements diagram are highly complementary covering many of the purposes of specifying requirements for RTS.

As discussed along this article, the main goal of the proposed approach is to combine and to apply SysML Requirements diagram with MARTE stereotypes, which enables modeling of individual software requirements for RTS. This article addresses this gap, as few approaches were proposed with focus on applying MARTE and SysML together to design RTS, and even less with the purpose of applying to requirements modeling. The focus is to create an approach for modeling specific software requirements which allows the representation of the necessary features of RTS (based on research demonstrating the specific requirements) and, also, timing requirements and performance requirements. A tool was developed to support the design of models. As SysML Requirements diagram is an extension of the UML Class diagram, the tool is an extension of the ArgoUML tool by using code clone techniques.

Expressing each requirement separately is highly desirable. This feature is addressed in this article by modeling requirements using the SysML Requirements diagram and by organizing the relationships between requirements. As each requirement is described separately, the complexity of changes is minimized, since a change in any requirement can be made completely and consistently maintaining the structure and style of the set of requirements.

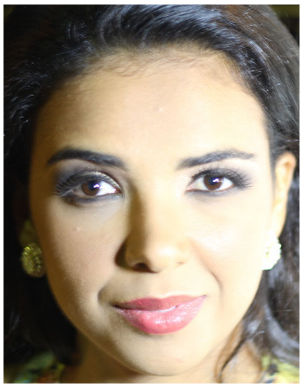
MARTE profile provides, in the proposed representation, a clear description of the various relevant aspects of requirements definition of RTS, as for instance, temporal aspects and constraints. Thus, in addition to the factors mentioned above, we can say that the proposed approach has an important role in approaches that specify RTS at different levels of detail and levels of abstraction.

Acknowledgements The authors would like to thank CNPq (www.cnpq.br) Grant 445500/2014-0 for the financial support.

References

1. Agarwal, R., Sinha, A.P.: Object-oriented modeling with UML: a study of developers' perceptions. Commun. ACM **46**(9), 248–256 (2003)
2. Albinet, A., Boulanger, J.-L., Dubois, H., Peraldi-Frati, M.-A., Sorel, Y., Van, Q.-D.: Model-based methodology for requirements traceability in embedded systems. In: Proceedings of 3rd European Conference on Model Driven Architecture Foundations and Applications (2007)
3. Argo (2015). http://argouml.tigris.org/
4. Bair, L.J., Tolk, A.: Towards a unified theory of validation. Proc. Simul. Conf. **2013**(Winter), 1245–1256 (2013)
5. Bantegnie, E.: INTERoperable embedded systems toolchain for enhanced design, prototyping and code generation. In: http://www.interested-ip.eu/project-description-objectives.html (2011)
6. Belategi, L., Sagardui, G., Etxeberria, L.: MARTE mechanisms to model variability when analyzing embedded software product lines. In: 14th Proceedings of the International Conference on Software Product Lines, pp. 466–470 (2010)
7. David, P., Idasiak, V., Kratz, F.: Reliability study of complex physical systems using SysML. Reliab. Eng. Syst. Saf. **95**, 431–450 (2010)
8. Demathieu, S., Thomas, F., Andr, C., Grard, S., Terrier, T.: First experiments Using the UML Profile for MARTE. In: 11th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2008), pp. 50–57. IEEE Computer Society, Silver Spring, MD (2008)
9. Dhavleesh, R., Rajesh, B., Maninder, S.: Software clone detection: a systematic review. Inf. Softw. Technol. **55**(7), 1165–1199 (2013)
10. Dubois, H., Peraldi-Frati, M., Lakhal, F.: A Model for requirements traceability in a heterogeneous model-based design process: application to automotive embedded systems. In: 15th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS), pp. 233–242 (2010)
11. Ebeid, E., Fummi, F., Quaglia, D., Stefanni, F.: Refinement of UML/MARTE models for the design of networked embedded systems. In: Europe Conference & Exhibition Design, Automation & Test, pp. 1072–1077 (2012)
12. Ebeid, E.S.M., Fummi, F., Quaglia, D.: Model-driven design of network aspects of distributed embedded systems. IEEE Trans. CAD Integr. Circuits Syst. **34**(4), 603–614 (2015)
13. Espinoza, H., Richter, K., Gérard, S.: Evaluating MARTE in an industry-driven environment: TIMMOś challenges for AUTOSAR timing modeling. In: Conference on Design, Automation and Test in Europe (DATE), MARTE Workshop, volume 07002 (2008)
14. Espinoza, H., Cancila, D., Selic, B., Gérard, S.: Challenges in combining SysML and MARTE for model-based design of embedded systems. In: Paige, R., Hartman, A., Rensink, A. (eds.) Model Driven Architecture—Foundations and Applications. Lecture Notes in Computer Science, vol. 5562, pp. 98–113. Springer, Berlin (2009)
15. Fishwick, P., Brailsford, S., Taylor, S. J. E., Tolk, A., Uhrmacher, A.: Modeling for everyone: emphasizing the role of modeling in stem education. In: Proceedings of the 2014 Winter Simulation Conference, WSC '14, pp. 2786–2796 (2014)
16. Gomez, C., DeAntoni, J., Mallet, F.: Multi-view power modeling based on UML, MARTE and SysML. In: EUROMICRO-SEAA, pp. 17–20 (2012)
17. Heisel, M., Cote, I.: A UML profile and tool support for evolutionary requirements engineering. In: 15th Software Maintenance and Reengineering, pp. 161–179 (2011)
18. Helming, J., Schneider, F., Haeger, M., Kaminski, C., Bruegge, B., Berenbach, B.: Towards a unified requirements modeling language. In: 15th International Workshop on Requirements Engineering Visualization (REV), pp. 53–57 (2010)
19. IEEE Std 830-1998. IEEE recommended practice for software requirements specifications, pp. 1–40 (1998)
20. Iqbal, M.Z., Arcuri, A., Briand, L.: Code generation from UML/MARTE/OCL environment models to support automated system testing of real-time embedded software. Technical Report 2011-04, Version 2, Simula Research Laboratory, Technical Report (2011-04), Version 2 (2011)
21. Iqbal, M.Z., Ali, S., Yue, T., Briand, L.: Experiences of applying UML/MARTE on three industrial projects. In: Proceedings of the 15th International Conference on Model Driven Engineering Languages and Systems, MODELS'12, pp. 642–658 (2012)
22. Kumar, B., Jasperneite, J.: UML profiles for modeling real-time communication protocols. J Object Technol. **9**, 178–198 (2010)
23. Laplante, P.: Real-time Systems Design and Analysis, 3rd edn. Wiley India Pvt. Limited, New York (2006)
24. Li, L., Ma, L., Wang, N., Yang, Q.: Modeling method of military aircraft support process based SysML. In: International Conference in Reliability, Maintainability and Safety (ICRMS), pp. 1247 – 1251. (2011)
25. Linhares, R.S., Oliveira, R.S., Farines, J.M., Vernadat, F.: Introducing the modeling and verification process in SysML. In: Conference on Emerging Technologies and Factory Automation, pp. 344–351 (2007)
26. Marques, M. R.S., Siegert, E., Brisolara, L.: Integrating UML, MARTE and SysML to improve requirements specification and traceability in the embedded domain. In: 12th IEEE International Conference on Industrial Informatics (INDIN), pp. 176–181 (2014)
27. Meng, C., Kim, S., Son, Y.-J., Kubota, C.: A SysML-based simulation model aggregation framework for seedling propagation system. Proc. Simul. Conf. **2013**(Winter), 2180–2191 (2013)
28. Mura, M., Murillo, L.G., Prevostini, M.: Model-based design space exploration for RTES with SysML and MARTE. In: Forum on Specification, Verification and Design Languages (FDL), pp. 203–208. (2008)

29. OMG, M.: UML Profile for MARTE: Modeling and Analysis of Real-time Embedded Systems Version, 1.1. Technical report, OMG. (2011a)
30. OMG, S.: Systems modeling language (sysml) specification—version 1.1. Technical report (2010)
31. OMG, U.: Unified modeling language—version 2.3. Technical report. version 2.3. (2011b)
32. Parviainen, P., Tihinen, M., Lormanms, M., Solingen, R.: Requirements engineering: dealing with the complexity of sociotechnical systems development. IdeaGroup Inc. Chapter 1. (2004)
33. Pinheiro, F.: Perspectives on Software Requirements, Chapter Requirements Traceability, pp. 93–113. Springer, Berlin (2003)
34. Quadri, I.R.: MARTE based modeling approach for partial dynamic reconfigurable FPGAs. In: Workshop on Embedded Systems for Real-Time Multimedia, pp. 47–52 (2008)
35. Quadri, I.R., Soares, L., Gray, I., Indrusiak, L.S., Bagnato, A., Sadovykh, A.: MADES: a SysML/MARTE high level methodology for real-time and embedded systems . In: 7th International conference on High-Performance and Embedded Architectures and Compilers, pp. 1–2 (2012a)
36. Quadri, I.R., Brosse, E., Gray, I., Matragkas, N.D., Indrusiak, L.S., Rossi, M., Bagnato, A., Sadovykh, A.: MADES FP7 EU Project: effective high level SysML/MARTE methodology for real-time and embedded avionics systems. In: 7th International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), pp. 1–8 (2012b)
37. Ribeiro, F.G.C., Soares, M.S.: An approach for modeling real-time requirements with SysML and MARTE stereotypes. In: ICEIS 2013—Proceedings of the 15th International Conference on Enterprise Information Systems, vol. 2, pp. 70–81 (2013)
38. Robbins, J.E., Hilbert, D.M., Redmiles, D.F. : Theories, methods and tools in program comprehension: past, present and future. In: 13th International Workshop on Program Comprehension (IWPC), pp. 181–191 (2005)
39. Roger, P.R., Elena, S.P., William, R.M.: Traffic Engineering, 3rd edn. Prentice Hall, New Jersey (2003)
40. Shousha, M., Briand, L.C., Labiche, Y.: A UML/MARTE model analysis method for uncovering scenarios leading to starvation and deadlocks in concurrent systems. IEEE Trans. Softw. Eng. **38**(2), 354–374 (2012)
41. Shuman, E.A.: Understanding executable architectures through an examination of language model elements. In: Proceedings of the 2010 Summer Computer Simulation Conference, SCSC '10, pp. 483–497 (2010)
42. Silvestre, E.A., Soares, M.S.: Modeling road traffic signals control using UML and the MARTE profile. In: 12th Computational Science and Its Applications (ICCSA 2012), pp. 1–15 (2012)
43. Soares, M.S.: A framework for multi-layered requirements documentation and analysis. In: Computer Software and Applications Conference (COMPSAC), pp. 308–313 (2011)
44. Soares, M.S., Vrancken, J.: Requirements specification and modeling through SysML. In: International Conference on Systems, Man and Cybernetics, pp. 1735–1740 (2007)
45. Soares, M.S., Vrancken, J.: Model-driven user requirements specification using SysML. J. Softw. **3**, 57–69 (2008)
46. Som, S.S.: Supporting use case based requirements engineering. Inf. Softw. Technol. **48**, 43–58 (2006)
47. Valles-Barajas, F.: A Formal model for the requirements diagrams of SysML. IEEE Latin Am. Trans. **8**(3), 259–268 (2010)
48. Von, B.M., Braun, P., Schroder, C.: Model based requirements engineering for embedded software. In: 10th IEEE International Requirements Engineering Conference, p. 92 (2002)
49. Wilde, N., Buckellew, M., Page, H., Rajlich, V., Pounds, L.: A comparison of methods for locating features in legacy software. J. Syst. Softw. **65**, 104–114 (2003)
50. Xu, J., Li, T., Xie, Z., Gao, T.: Use cases and feedback in functional requirements analysis. In: Information Technology, Computer Engineering and Management Sciences (ICM), vol. 2, pp. 54–57 (2011)
51. Zaki, M. Z.M., Jawawi, D.N.A.: Model-based methodology for implementing marte in embedded real-time software. In: IEEE Symposium on Computers & Informatics (ISCI), pp. 536–541 (2011)



Fabíola Gonçalves C. Ribeiro received her B.Sc. in Computer Science from Federal University of Goiás (UFG) in 2008 and her M.Sc. in Computer Science from Federal University of Uberlândia (UFU) in 2013, both in Brazil. Since 2014 Mrs. Ribeiro is a Ph.D. student in Computer Science from Federal University of Uberlândia (UFU). Her research interests include specification and design of embedded and real-time systems using UML profiles (SysML and MARTE), with focus on Intelligent Transportation Systems.

Carlos E. Pereira is a Full Professor at Federal University of Rio Grande do Sul (UFRGS) in Porto Alegre, southern Brazil, where he chairs the Control, Automatic, and Robotics Group, a research group. He received the Dr.-Ing. degree in Electrical Engineering from the University of Stuttgart, Germany, in 1995 and has a MSc in Computer Science and a B.S. degree in Electrical Engineering both from UFRGS. He is currently acting as Director of Operations at EMBRAPII in Brazil. He is a researcher level 1 of the Brazilian Research Agency CNPq and has more than 400 technical publications on conferences and journals. He is Associate Editor of the Journals "Control Engineering Practice" and "Annual Reviews in Control" from Elsevier. His research focuses on methodologies and tool support for the development of distributed real-time embedded systems, with special emphasis on automation applications and the use of distributed objects over industrial communication protocols. He has also worked on several research projects in collaboration with industry, mostly dealing with the development of real-time computer-based systems and applying the concepts developed on his scientific research. In 2012 Prof. Pereira received the Friedrich Wilhelm Bessel research award from the Alexander von Humboldt Foundation.

Achim Rettberg was born in Einbeck, Germany. He received his M.S. (Dipl.-Inform.) in computer science and economics in 1997 and his Ph. D. (Dr. rer. nat.) degree in 2006 from the University of Paderborn, Germany, where he studied computer science and economics. From 1992 until 1997 he was employed as a working student for the University of Paderborn and C-LAB. From 1997 until 2000 he worked at C-LAB in industrial projects together with dSPACE and Siemens. From 2001 until 2006 he worked as a Ph.D. student and researcher for the C-LAB in the field of embedded system design. Afterward he worked from 2007 until 2008 as Postdoc at C-LAB and managed EU Projects. In 2008 he became professor of complex integrated systems/embedded systems at the Computer Science department at Carl von Ossietzky University Oldenburg. Prof. Rettberg is founder and General Chair of the International Embedded Systems Symposium (IESS). Furthermore, he is active in several conference/workshop committees and journals. Prof. Rettberg is author and co-author of several scientific books, journal articles and conference papers. His general research interests are model-based design, real-time systems and HW/SW architectures. He is chairman of the IFIP working group 10.2 for embedded systems.

Michel S. Soares received his B.Sc. in computer science in 2000 from Federal University of São Carlos and his MSc in computer science in 2004 from Federal University of Uberlândia, both in Brazil, and a Ph.D. with focus on software engineering in 2010 from Delft University of Technology in The Netherlands. He has worked on several research projects in collaboration with industry and government. Currently, he is an Assistant Professor at Federal University of Sergipe in São Cristóvão, Brazil. His research interests include software architecture, model-driven systems engineering, requirements engineering, service-oriented architectures and software quality. Dr. Soares has published more than 70 technical papers in scientific journals, conferences and books.