A Model-Based Engineering Methodology for Requirements and Formal Design of Embedded and Real-Time Systems

Fabíola Gonçalves C. Ribeiro*, Achim Rettberg†, Carlos E. Pereira‡, Michel S. Soares§
*Federal Institute Goiano, Catalão, Brazil
Email: fabiola.ribeiro@ifgoiano.edu.br
†Carl von Ossietzky Universitt Oldenburg, Oldenburg, Germany
Email: achim.rettberg@iess.org
‡Federal University of Rio Grande do Sul, Porto Alegre, Brazil
Email: cpereira@ece.ufrgs.br
§Federal University of Sergipe, São Cristóvão, Brazil
Email: mics.soares@gmail.com

Abstract—**Activities for the comprehension and development of Cyber-Physical Systems (CPS) include analysis of multiple disciplines including mechanical engineering, electronic engineering, systems engineering and computer science. This work presents a comprehensive and applicable methodology for the initial activities of the development process of CPS. This methodology displays the capacity to describe and enable detailed analysis of the relevant properties of these systems as, for example, time specification, resources, communication and non-functional properties of CPS. In this research, two consolidated approaches of Model-Based Engineering are used in a combined way for proposing a methodology for requirements analysis, modeling and formal specification of CPS. Initially, a strategy for the definition, modeling, specification, and categorization of requirements in a tabular way is proposed. From the system definition in a high abstraction level, the SysML Requirements diagram is extended by using UML profile MARTE/VSL for formalization of restrictions, annotations and stereotypes in the model. Initial results of the application of the proposed methodology are presented by means of a case study of the Industrial Packing System.**

1. Introduction

Currently, manufacturing systems have a high degree of automation, which increases complexity during their development. These systems are used to control physical and logical components, and also have to consider quality aspects such as efficiency and reliability. Among the objectives of such systems in industry one can mention the need for cost reduction of traditional production processes, the ability to create distributed control in an agile manner, and the high need for customization and fast reaction regarding demands of consumers.

Due to operational capacity and high intelligibility, many manufacturing systems are also considered Cyber-Physical Systems (CPS) [1]. CPS are characterized as intelligent systems that are composed of digital virtual/cyber technologies, software, and physical components, and intelligently interact with other systems across information and physical interfaces [2]. These systems are able to interact with external environment and generate immediate responses to the environment. Analysis and specification approaches are important, in this context, for enabling definition of the main functionalities, restrictions and response times imposed on these systems [3].

Model-Based Engineering is an approach for analysis, specification and design of systems that intend to raise the level of abstraction through the use of models in systems development activities. MBE can be seen as a trend in the design of smart automation systems. Several MBE approaches have been proposed in recent years as, for example, in the field of automotive systems [4], [5], traffic control systems [6], [7], unmanned aerial vehicle systems [1], among others.

MBE approaches can be employed under different perspectives and interests, allowing one to analyze and describe mechanical components which are usually integrated with electronic and software components [8], as well as providing high relevance to handle semantic gap between specification and system design and the actual features of an application. For instance, in [9], the MBE approach has been applied in case studies at Renault automotive systems to structure the processes and activities of architectural design. A methodology for integrated design of mechatronic systems is presented in [10], in which the W model and SysML are employed to compose the design methodology that will be evaluated on a case study of the filling system of a Tetra Pak Packaging Solution. MBE is applied in [11] to define the functional specifications, systems modeling, the traceability criteria for an automotive drive line system, with the benefit of providing a wide communication process through a common language. In [2], MBE approaches are used to describe a production cell of the SmartFactory System. In the case study, the assembly modules, the material identification and the flow control are presented using SysML Blocks diagram. Another example of using SysML is described in [12], in which SysML is extended resulting in SysML-AT (SysML

URI: http://hdl.handle.net/10125/41905
ISBN: 978-0-9981331-0-2

for automation), which describes the hybrid characteristics of a manufacture system focusing in the centralized, decentralized and distributed hardware architectures. An aspect-oriented MBE approach, named as Aspect-Oriented Model Driven Engineering for Real-Time systems (AMoDE-RT), is proposed in [8]. AmoDE-RT is applied in the design of industrial mechatronic systems, more specifically for controlling a product assembler industrial cell.

Manufacturing systems are also characterized by great complexity of their physical and logical components and, especially, cooperation and communication between heterogeneous subsystems. Therefore, it becomes essential the adoption of good practices and processes to deal with representation and definition of different properties of these systems in order to achieve greater integration and consistency. Non-functional nature of some important requirements of embedded systems, such as dispersion and confusing manipulation of their properties can lead to several problems about their development, comprehension and deployment. If these problems are not treated properly, the overall project complexity regarding spending efforts and project chronogram can be increased [8]. In the design phase it is important to raise the level of abstraction in the early stages of system development cycle in order to facilitate the understanding and analysis of different embedded components and their relationships by most stakeholders. Models at high abstraction levels minimize possible omissions and complexity [13].

Research presented here presents a methodology for the initial activities of the development process of CPS, where new methods/profiles, suitable for CPS domain. This research differs from the previously mentioned works due to three main reasons. The proposed MBE approach aims to integrate a SysML based design for specification of complex systems, with domain-specific concepts of CPS (by using MARTE profile constructors). The proposed approach aims to be initially tailored for analysis processes, textual and graphical requirements specification which are relevant to real-time systems. Finally, for allowing future validation and verification processes, formalized descriptions are employed in modelling elements for the non-functional requirements of the IPS system. The methodology proposed in this paper is applied to the design of industrial mechatronic systems with focus on analysis, specification and design of embedded and real-time characteristics of CPS.

2. Brief Introduction to Value Specification Language

Within VSL, semantics and syntax is presented to describe DataTypes (primitive DataTypes and enumeration), literal value (real literal, DateTime and default literal), expressions, composite values (for example, interval collections, tuples and choices), and also expressions of time, which allows specifying temporal values and expressions.

VSL extends the UML simple time model by adding capabilities to improve the properties for a description of different data types, provides criteria for specification of literal constant values, offers support to describe different types of expressions in order to allow the description of references to variables, and allows to represent different types of expressions. In addition, VSL describes, through package TimeExpressions, a specialized syntax for writing expressions and specifications of time values in the model elements.

Concepts presented above characterize abstract syntax and semantics of VSL. A concrete syntax is described by means of the value specifications for an expression, being the construction of an expression as shown in expression 1:

$$\begin{aligned}<value-specification>:: \quad & =<literal>| \\ & =<interval>| \\ & =<enum-specification>| \\ & =<collection>| \\ & =<tuple>| \\ & =<choice>| \\ & =<expression>| \\ & =<time-expression>| \\ & =<obs-call-expression>\end{aligned} \tag{1}$$

Value Specifications are used to specify the textual value parts of UML models. The value specification could be a simple literal, such as a number, or it could be a complex expression that involves variables and operations. A full form of atomic definition for formulation of expressions is presented in expression 1. Understanding each value specifications becomes important to define functional constraints modeled elements, as well as to formulate more complex temporal expressions which usually groups one or more expressions of a value specification.

Time Expression, presented in expression 2, allows to formalize different temporal and non-functional expressions on elaborated models and are important for providing different and rigorous standards for representation of expressions. In general, a **Time Expression** enables description of intervals (minimum and maximum) and duration of an event, as well as the distance considered between consecutive events (see Expressions 9 and 10), to make explicit the specific moments of occurrence of an event (see Expression 7), detail specific event durations (observe Expression 3), describe occurrence of conditional events and, also, specify possible variations in events that can be represented in model elements (Expression 8).

$$\begin{aligned}<time-expression>::= \quad & <duration-expr>| \\ & <instant-expr>| \\ & <jitter-expr>\end{aligned} \tag{2}$$

Definition 1: *A **DurationExpression** is a temporal expression that evaluates as a duration value for the event represented. Its syntax is defined by expression 3:*

$$\begin{aligned} <duration-expr>::= \quad & (<real-literal>|variable-call-expr>)| \\ & <duration-obs-expr>| \\ & ('('<instant-obs-expr>'-' \\ & <'instant-obs-expr>')') \end{aligned} \tag{3}$$

A Duration Expression can also be decomposed into the following expressions 4, 5 and 6:

$$<variable-call-express>::= \quad <variable-name>| \tag{4}$$

$$\begin{aligned} <duration-obs-expr>::= \quad & <duration-obs-name> \\ & ['['<occur-index-expr>']']['when' \\ & <condition-expr>']'] \end{aligned} \tag{5}$$

$$\begin{aligned} <instant-obs-expr>::= \quad & <instant-obs-name> \\ & ['['<occur-index-expr>']']['when' \\ & <condition-expr>] \end{aligned} \tag{6}$$

Specifications presented in expressions 4, 5 and 6 need definition of the following atomic values for an expression:

$$\begin{aligned} <variable-name>::= \quad & [<namespace>'.']<body-text> \\ <duration-obs-name>::= \quad & [<namespace>'.']<body-text> \\ <instant-obs-name>::= \quad & [<namespace>'.']<body-text> \\ <occur-index-expr>::= \quad & <value-specification> \end{aligned}$$

Definition 2: *: An **InstantExpression** allows to describe temporal expressions that are able to represent a value of instant of time. Its syntax is defined in Expression 7:*

$$\begin{aligned} <instant-expr>::= \quad & (<datatime-literal>|<variable-call-expr>) \\ ::= \quad & ['+'<duration-obs-expr>)| \\ ::= \quad & (<instant-obs-expr>['+'<duration-expr>]) \end{aligned} \tag{7}$$

One datetime-literal provides the following specifications:

$$\begin{aligned} <datetime-literal>::= \quad & (<data-string>[daystring]) \\ ::= \quad & |(<time-string>[data-string>] \\ ::= \quad & [day-string])|(<daystring>]) \end{aligned}$$

Definition 3: *: A **JitterExpression** describes a variation (jitter) in the occurrence of events that occur in instants separated by periodical intervals. Occurrence of these variations must be detailed to enable the control of possible variations or delays in events/actions in the system. Expression 8 describes the different possibilities for description of a jitter.*

$$\begin{aligned} <jitter-expr>::= \quad & ('jitter('<instant-obs-expr>')')| \\ ::= \quad & ('jitter('<instant-obs-expr> \\ ::= \quad & '-'<instant-obs-expr>')') \end{aligned} \tag{8}$$

Definition 4: *Expressions that refer to representation of temporal instants or event durations to be expressed are presented in expressions 9 and 10. Both expressions describe a special type of interval specification that has temporal expressions with lower and upper limits:*

$$\begin{aligned} <instant-interval>::= \quad & ('['|']')<instant-expr>'..' \\ ::= \quad & <instant-expr>('['|']') \end{aligned} \tag{9}$$

$$\begin{aligned} <duration-interval>::= \quad & ('['|']')<duration-expr>'..' \\ ::= \quad & <duration-expr>('['|']') \end{aligned} \tag{10}$$

Often, when specifying real-time systems, one needs to represent time cardinality (delays, duration of events, clock time, chronometric time, and models to represent logical time).

Each one of these expressions are applied in a rigourous way for specification of models. Therefore, abstractions, restrictions, requirements and models specifications, which are most often described in natural language, can be formally specified using VSL expressions. As a matter of fact, representation of temporal expressions will be used to different kinds of time related expressions, including instants, durations, and jitters.

3. Proposed Approach

Fig. 2 presents the proposed approach in order to establish several contributions in the real-time and embedded domain. This approach supports the specification process and documentation requirements, requirements analysis, requirements consistency check (as future work), and also contribute to activities of system design. The main focus is in the development process of systems with embedded and real-time features.

Initially, different domain characteristics are presented and analyzed in order to find out the different functional modules that represent the system or component in question. In this sense, in this activity the initial needs of the system must be conceived, analyzed and listed in natural language.

SysML Requirements diagram is further elaborated to enable requirements elicitation (stage of initial requirements

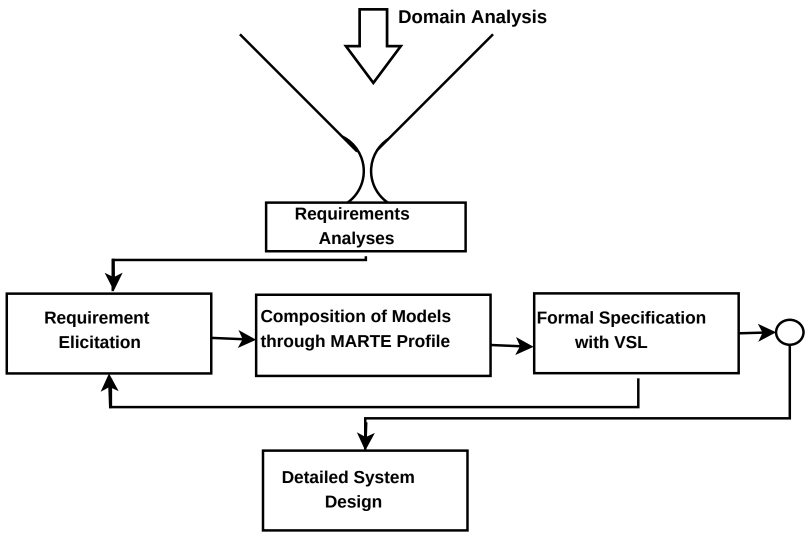

Figure 1. Proposed Approach

engineering process) of the proposed system. Initial models are constructed by means of SysML Requirements diagram, allowing to detail the different requirements that make up the system and their relationships and the different compositions and dependencies between one or more requirements.

As previously presented, CPS have specific characteristics that should be clearly specified in order to avoid development omissions. Therefore, constructors from the MARTE profile are included in initial requirements specification models to explicit temporal requirements (physical and logical), non-functional and, also, dynamic and static criteria that are relevant to a requirement.

The final stage established by the proposed approach, for the processes of requirements engineering, describes the refinement of developed models and all the descriptions and annotations in the model elements through VSL. This step aims to establish formal annotations of model elements, which facilitates the analysis, to contribute to activities of the design process (such as refinements) and, mainly, to allow one to perform activities that verify the correctness and consistency in a requirements specification.

The steps presented above are considered for description of the structural components of the system (briefly introduced in Section 4), their relationships and internal components. Detailed design artifacts are presented in Sections 5, 6 and 7.

4. Case Study and Application of the Proposed Approach

Industrial Packing System (IPS) is an industrial mechatronic system that owns different embedded and real-time functions employed in industrial processes for different purposes, degrees of complexity and automation. Figure 2 depicts diagrammatically system components used for this case study, which has been adapted from other proposals [14], [15] [8].

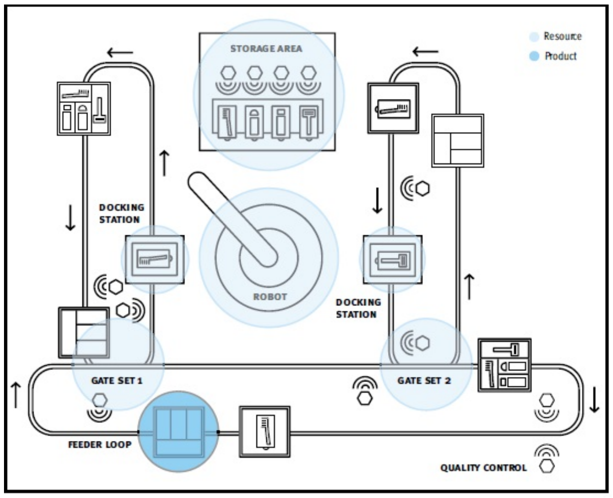

Figure 2. Industrial Packing System. Adapted from [15]

IPS is composed of the following components: feeder circuit that performs routing of items, boxes and packaging boxes, by the conveyor belt, between the stations and the control area, two gate sets that allows to perform routing of items, boxes and packed boxes between the conveyor loops, two conveyor belts providing routing of items, boxes and packed boxes between transport loops, a robotic arm with a gripper that owns movement operations of materials for storage unit or for mounting, a storage unit to store items for assembly or packages/parts already packaged, tag readers which aim to check information of a piece and decision making from it, and various sensors which jointly indicate conditions on components, including items, boxes and parts, handled by the system.

The input conveyor belt brings individual parts that are combined to form products. The conveyor belt stops when the sensor detects presence of a part. Then, the robotic arm will put it in a unit of storage. The second conveyor belt brings empty boxes in which the parts are stored. This conveyor belt continues to operate until the sensor detects an empty box in the expected position. When the product is fully assembled, the controller sends a command to the conveyor belt which must initiate the product packaging and subsequent storage.

When a new product requires a part or item, the status of sensors coupled in the assembly conveyor is evaluated to check whether the product is available. If the item is physically located in the parts of the conveyor belt, the product will be assembled from this conveyor belt and the available item used to assemble the product. Otherwise, the piece is taken from the storage unit.

Functional and non-functional requirements of IPS are analyzed, elicited and specified below, according to the proposed methodology. SysML and MARTE profiles are adopted for generating structural models of the system in order to describe embedded and real-time properties of requirements and system design models.

5. Requirements Analysis

In order to apply the modeling methodology elaborated in this research (Section 3), the initial description of IPS is evaluated and the analysis process of its main features is initialized with the purpose of developing, at a high level of abstraction, a check list that matches the key responsibilities of the system. Each one of the requirements is tabulated, as presented in Table 1, according to its type. Non-functional characteristics correlated and the indication of which elements, of the MARTE profile, can contribute to the design approaches (next steps of the proposed approach) is also shown in Table 1. This analysis stage, categorization and description in high level of abstraction of requirements allows better understanding of these requirements and, later, formal specifications.

As discussed in other works [16], [8], non-functional requirements of a real-time system may be contained in four major categories, being them time, performance, distribution and embedded.

Time Requirements may be related to the definition/fulfillment of *timing* or *precision* criteria. Among the timing requirements can be found non-functional requirements related to *deadlines*, *periods*, *cost*, *time release*, *activation latency* and *deadlines start and end of activities*. Furthermore, requirements that are related to the type *precision* can be of type *tolerated delay*, variation in the execution of activities (*Jitter*), *freshness* (represents the time interval in which a sampled data value is considered to be updated), *resolution* and *drift*, which has a derivation or continuous displacement that must be shown.

Performance category is related to the definition of criteria regarding *response time* and *throughput* of a non-functional requirement. These characteristics are important for explicit performance criteria that must be well-defined in order to create effective and secure systems.

Distribution category is composed of requirements that can describe characteristics of *allocation of tasks* in different distributed modules, information about the different *hosts* of the system, and decentralized *communication* and *synchronization* policies between the different modules dispersed of the system. Finally, the relevant characteristics related to embedded category may be related to *area*, *power consumption*, *total energy* and *memory allocation* of an embedded component.

Table 1 describes initial system analysis towards the high-level requirements of IPS. In general, this analysis is very important for the initial process definition and understanding of the system and the separation of concepts that relates the functional, non-functional and embedded properties of the system. Requirements for high system level are described in natural language (column Requirement Description), their categorization in terms of functional and non-functional requirements (column Type), the description of the subcategories and, also, with this analyses it becomes possible to provide, for specification purposes or detailed design, the MARTE profile packages that will detail this requirement.

6. Formal Specification of Requirements with MARTE and VSL

SysML Requirements diagrams [17] provide support for modeling individual requirements and their relationships and makes it possible to represent a requirement in an atomic manner. Each individual SysML requirement contains two attributes named as **ID**, a unique identification, and **TEXT**, to describe general information about the requirement.

In this proposed research, SysML Requirements diagram is adopted in each of the phases of the approach proposed in Section 3, at different levels of detail, regarding activities of Requirement Elicitation, Composition of Models through MARTE Profile and Formal Specification with VSL.

Activities of Requirement Elicitation use the related information about global requirements of the Industrial Packing System, presented in Table 1, for modeling the system requirements and explain the relationships and the cooperation between general and more specific requirements of the system. This early analysis contributes to understanding the purpose of the system. Also, it allows to establish its different modules, shows the hierarchy and dependencies between system requirements and cooperate to define the artifacts in the system design stage. Fig. 3 presents a fragmentof process requirements elicitation. The main objective of requirements elicitation is to present models of different system requirements at a high abstraction level and show how these requirements are correlated or have influence on other system requirements.

From the directions on non-functional and embedded characteristics, presented in Table 1, Composition of Models through MARTE Profile aims to explicit in the elaborate models different annotations, stereotypes or restrictions on model elements. Elements modeled in this stage are

Requirement Description	Type	RFN Type	MARTE Package
The system must control the robotic arm	RF	-	NFP
The system must control the mechanical gripper of moving objects	RFN	Time/Precision	Time/GRM
The system must control the mechanical joints arm control	RFN	Time	Time
The system must control the assembly cell	RFN	Time/Distribution	NFP/Time/GRM
The system must control the product packaging	RFN	Time/Distribution	NFP/Time/GRM
The system must control the storage unit	RFN	Time/Distribution	NFP/Time/GRM
The system must control the movement of the conveyor belt	RF	-	NFP
The system must perform the detection of partes in the conveyor belt	RFN	Time(Period/Freshness)	Time/GRM
The system must perform the detection boxes in the conveyor belt	RFN	Time(Period/Freshness)	Time/GRM

TABLE 1. ANALYSIS AND CLASSIFICATION OF SYSTEM REQUIREMENTS IN HIGH LEVEL

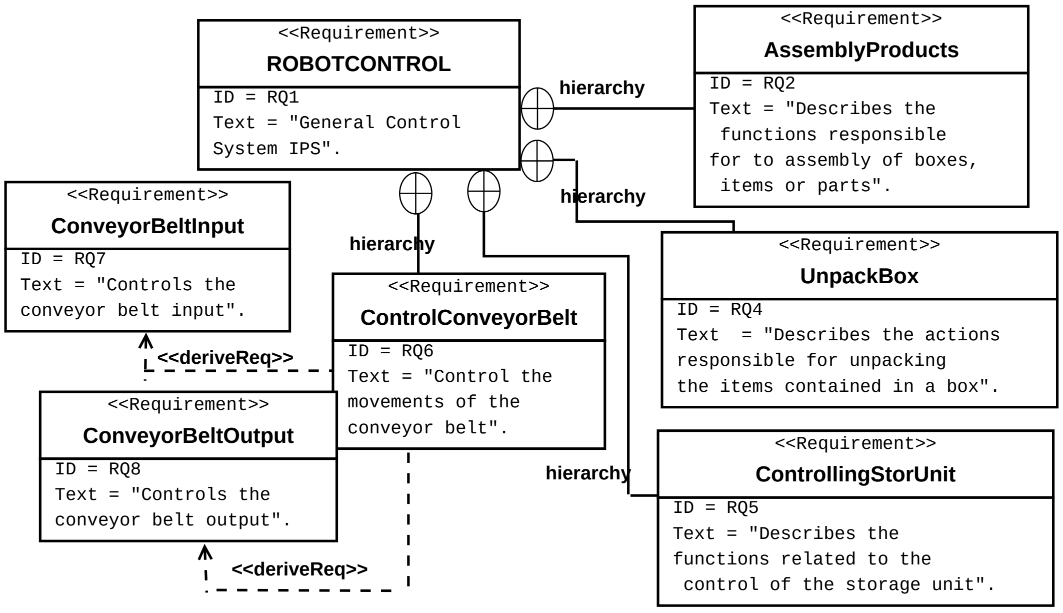
<<Requirement>>
ROBOTCONTROL
ID = RQ1
Text = "General Control System IPS".
<<Requirement>>
AssemblyProducts
ID = RQ2
Text = "Describes the functions responsible for to assembly of boxes, items or parts".
<<Requirement>>
ConveyorBeltInput
ID = RQ7
Text = "Controls the conveyor belt input".
<<Requirement>>
ControlConveyorBelt
ID = RQ6
Text = "Control the movements of the conveyor belt".
<<Requirement>>
UnpackBox
ID = RQ4
Text = "Describes the actions responsible for unpacking the items contained in a box".
<<Requirement>>
ConveyorBeltOutput
ID = RQ8
Text = "Controls the conveyor belt output".
<<Requirement>>
ControllingStorUnit
ID = RQ5
Text = "Describes the functions related to the control of the storage unit".
hierarchy
hierarchy
hierarchy
hierarchy
<<deriveReq>>
<<deriveReq>>

Figure 3. Part of IPS Elicitation Requirements

enriched with MARTE constructors allowing to describe non-functional criteria relating, for example, non-functional properties timing, accuracy, performance, embedded and distribution requirements and, also, the description of physical and logical resources of the system.

Finally, the last phase in the formulated approach proposes Formal Specification by describing models developed with VSL. VSL is a compatible language with the constructors of the MARTE profile and is used to allow formal specification, on model elements, of requirements with temporal characteristics and in annotations based on expressions. Thus, it becomes possible to verify the correctness and consistency of elaborated models with the proposed specification.

Due to space constraints, only the modeling of component **ROBOTCONTROL** of IPS is presented in this paper. Moreover, artifacts of phases 2 and 3 of the proposed approach are composed and presented together in Fig. 4. **ROBOTCONTROL** represents one of the most complex resources of IPS, possessing as main functions the communication, coordination and interaction with embedded components (e.g., robotic arm), physical (conveyor belt) and logical (processing units) of IPS.

Requirement **REQ1** keeps control and coordination of the four processing activities of the system. These activities, named as $A1$, $A2$, $A3$ and $A4$, represents, respectively, the conveyor belt input control, conveyor belt output control, the coordination (or handling) of assembling arms and control of the storage unit. This requirement is stereotyped with $<< Configuration >>$ ((*of CoreElements::Causality::ModalBehavior*) and is important for representing the main system configuration that is defined by a set of elements or modules of the system.

Requirement **RQ2** represents a non functional timed requirement whose actions performed in the system occur after sending a signal from element ROBOTCONTROL for this component. Packing of a box or of an item is accomplished based on interpretation of a signal (EVENTROBOT) sent by **ROBOTCONTROL** for this element. Therefore, the action of packing a box is triggered if $Nb = 1 * Ni = 0$. On the other hand, the action of assembling an item is performed if $Ni = 1 * Nb = 0$. The Nb variable refers to the number of boxes and Ni to the number of items which are associated with the EVENTROBOT event. Therefore, when a solicitation of a box arrives to the robot, an action of packing a box is triggered.

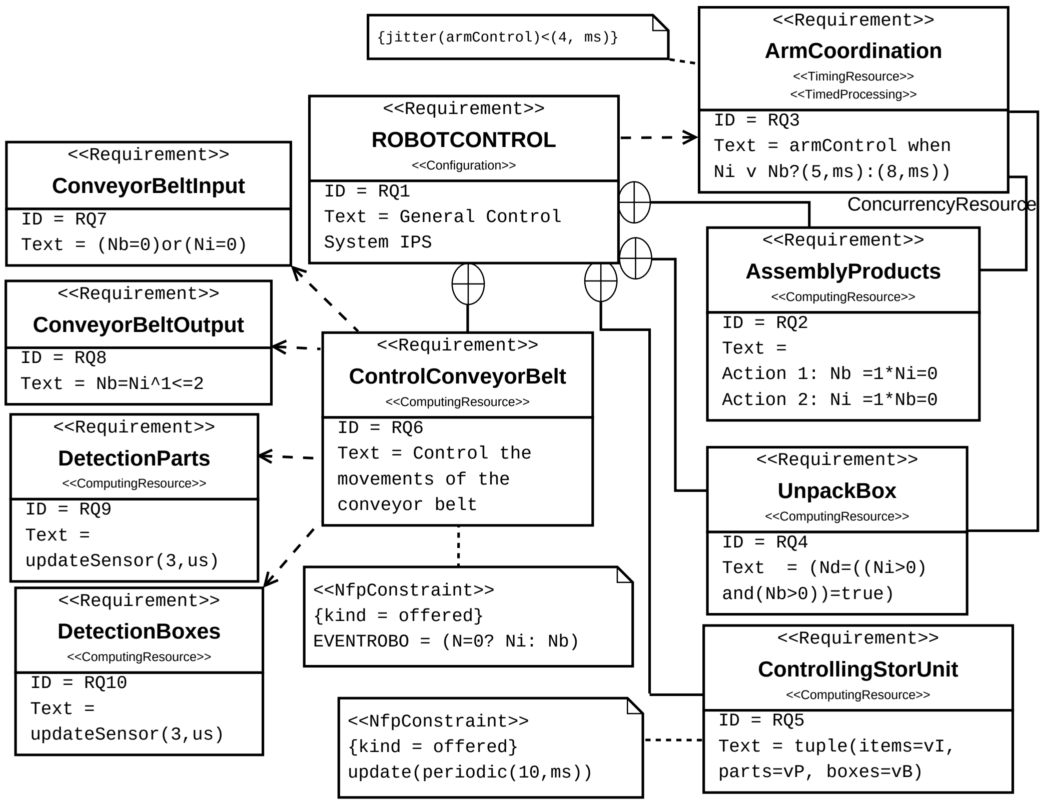

Figure 4. IPS Requirements Specification with MARTE and VSL.

Requirement **RQ3** describes a timed element, through two Time Expressions. Initially, a duration expression delimits the duration period for performing the EVENTARM, which is assigned from ROBOTCONTROL. In this case, the packed box and items of the arm operations have the duration of $3ms$. The unpacking operation has the duration of $8ms$. These durations are defined by the conditional expression contained in Text. For this element the maximum variation (Jitter) of $4ms$ for its controls/functions is defined. Stereotype $<< TimedProcessing >>$, from Time::TimeRelatedEntities::TimedProcessing Models, is applied in this requirement to indicate that the functions which are executed by RQ3 describes start and end times or a known duration. In addition, stereotype $<< TimingResource >>$, from MARTE::GRM::ResourceTypes, is also employed in this requirement to show that it will interact with hardware components that are capable of following and evidencing pace of time.

Requirement **RQ4**, when receiving an EVENTARM, will perform, through cooperation with $RQ3$, unpacking of items of a box in accordance to the analysis of boolean expression $Nd = (Ni > 0) \wedge (Nb >= 1) = true$. Requirements $RQ2$ and $RQ4$ relates with $RQ3$, once this requirement cooperates in a synchronized way with the above-mentioned requirements. Stereotype $<< ConcurrencyResource$, from MARTE:GRM::ResourceTypes, describes logical or physical concurrency. In case it is logical, the supplying processing resource needs to be arbitrated with a certain policy.

Besides keeping updated information of items, parts and boxes, requirement **RQ5** allows to perform classification of the storage unit items through tag reading. Requirement **RQ5** is described by an expression that matches with Tuple Type, that is, $tuple = (itens = vI, parts = VP, boxes = vB)$ in which vI, vP and vB variables describes a literal decimal number that corresponds to the updated values (periodically in this unity) of items, parts and boxes. Stereotype $<< NfpConstraint >>$ is used to write a $RQ5$ restriction, as a temporal expression, to describe the periodical execution of an update in the quantity of items from the storage unit.

Stereotype $<< ComputingResource >>$ is employed in the model elements that represents either virtual or physical processing devices that must be able to execute a program code to accomplish its functionalities.

Requirement **RQ6** controls movements of the conveyor belt. Requirements **RQ7** and **RQ8** relate control functionalities for conveyor belt input and conveyor belt output of products which are moved by conveyor belt. Boolean function $(Nb = 0) \vee (Ni = 0)$, expressed in requirement $RQ7$, checks either the nonexistence items or boxes in the input conveyor. If both are equal to zero, then it is necessary to remove such elements from the storage unit for the input conveyor belt. Data transference between the output conveyor belts and for the storage unit (transportation and stacking boxes) is controlled by **RQ6** from evaluation of boolean function $Nb = Ni^1 <= 2$ in requirement **RQ8**, which describes that no more than two boxes can be in the conveyor belt. Furthermore, these requirements are subjected to adverse situations presented by requirements **RQ9** and **RQ10**, and must provide a set of operations for acting accordingly to changes and a potential disorder during production (e.g., a sequence of interrupted products).

Requirements **RQ9** and **RQ10** represent functions responsible for evaluating, through periodical reading of sensors, conditions of the conveyor belt as it relates to the absence or existence of items (this will trigger a behavior on the assembly vinculated system) and/or boxes already assembled by the system. For avoiding delays in the other processes of the system such readings occurs in a restrict period of time and must have a low sampling tax.

VSL specifications allows to specify, in a clear way, in the requirements model, processing activities that must be timed or requiring a formal declaration of their requirements. By using VSL, it is possible to formalize intervals or durations that restrict a requirement, as for instance, conditional expressions in elements of the model and possible variations in specific actions.

7. Detailed System Design: Block Diagram

SysML Block diagram represents the system components and their relationships [17]. A block can represent the characteristics of a physical or logical entity, such as a hardware component or a physical object. Blocks can have attributes and operations and these represent, respectively, the internal properties of the block (shown in the second compartment) and operations that describe the behavior presented by the block (shown in the third compartment).

The Industrial Packaging System structural analysis is represented in this research through SysML structural diagrams enriched with MARTE annotations. Figure 5 depicts the SysML Block diagram for a part of the system modeled in this case study. The modeled system is shown partially due to lack of space in this paper, and by focusing on the components that are related to process control, process assembly/disassembly of a product, from the captured signals of Conveyor Belt, controlled by System Control and interpreted and executed by Assembly Module and Arm Device.

Detailed design of the system architecture is composed, in this partial model, by **System_Control**, which describes the interactions of an autonomous component able to self manage, interact and control the other components of the system, **Conveyor_Belt**, which describes the characteristics of the control module of the input and output conveyor belt, **Bus_Control**, responsible for configuring communication interface between components, and **Arm_Device**, component that characterizes an embedded physical device able to perform the processes of packing and unpacking an item or boxes in accordance with the requests for product assembly (RQ2), and **Assembly_Module**, which describes an element which binds events triggered in System_Control assembling operations Arm_Device.

The pack attribute, specific bus, allows to represent data packet transferred between the system components (signals which characterize the system event). EVENTROBOT and EVENTARM attributes identify events, captured by sensors,and behavior of the system. In addition, attributes are described that assign the ports/interfaces for communication between the system components as, for example, IPS_1, IPS_2, IPS_2 and IPS_4.

The third compartment of a block, depicted in Figure 5, describes the behavior function provided by a block. For example, operating of the System_Control, named as getInforConveyor(), describes the configuration of the system primary activities. That occurs from periodic reading sensor data that is initialized for executing the operation updateInfoConveyor(), which occurs in a synchronized way with operations of block Conveyor_Belt). Other operations, not described here for reasons of space, relate to relevant operations of provided interfaces or required of the block in question.

Figure 5 depicts communications between System_Control and the other blocks. The block diagram relates a block to another through associations that refers to possible communications/relationships between parts of a block. The composition relationship is represented by an arrow with a filled diamond at one end. This relationship establishes an all-part relationship in which the block at end of the diamond represents the block-all and is composed by blocks part. Thus, it is possible to make explicit that blocks Arm_Device, Conveyor_Belt, Assembly_Module and Bus_Control compose and integrate the System_Control block and Cooperate together for the attendance of the general purpose of the system.

Stereotypes used in the design model allow to emphasize resources, characteristics of physical entities, communication between elements, and other tasks. The stereotype CommunicationMedia allows to describe elements of the system capable of carrying information from one location to another. The elementSize attribute describes the size in bits of the transmitted message, and the transmMode attribute represents the transmission mode of packages.

The Device stereotype is used to characterize a physical element with one or more offers to the service system. Stereotype TimingResource is applicable, in design models, to emphasize an embedded device that has relationship with a chronometric clock in order to implement the timing mechanisms in their operations. In this case, the TimedEvent stereotype relates a model element to a logical or physical clock.

Scheduler and ProcessingResource describes a system component that performs its processing activities in accordance with certain scheduling policies, that is, from the ordering (through a pre-established policy) to the events that will be executed. Stereotype TimedEvent enables events that should happen in known time intervals. For the Conveyor_Belt block this stereotype is pertinent to describe the time-verification process carried out by the sensors

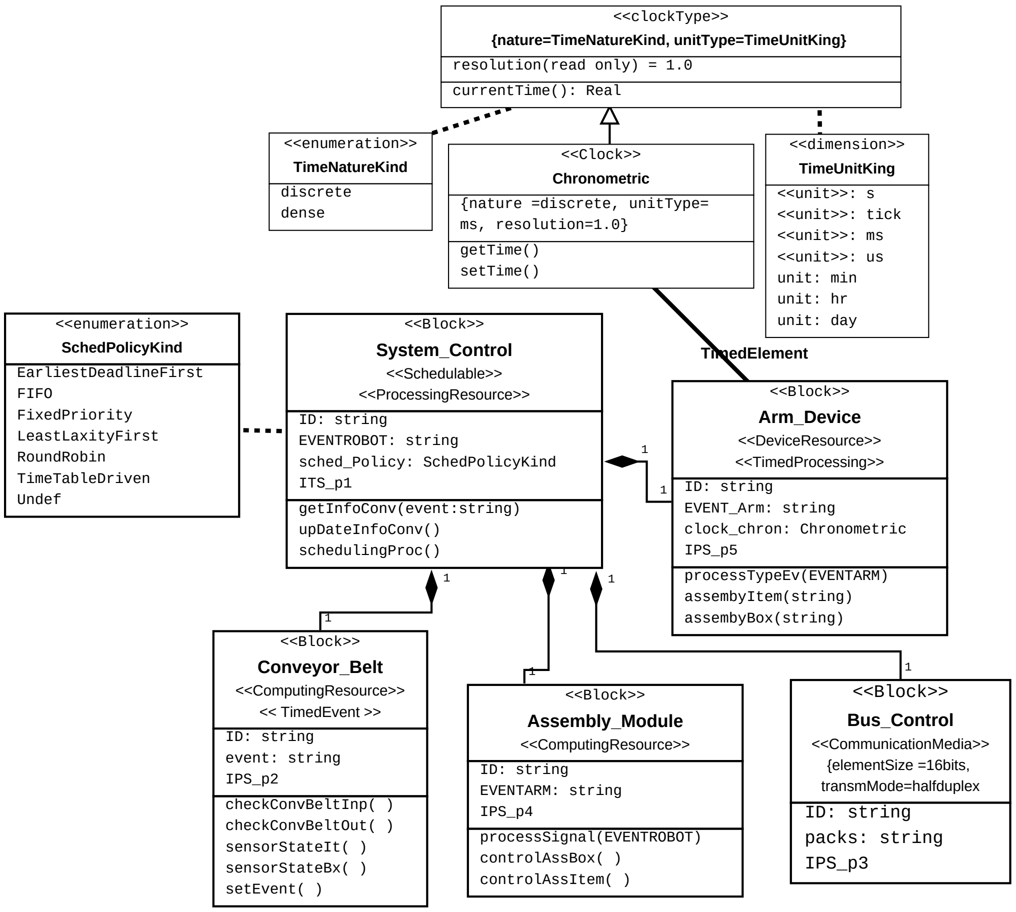

Figure 5. Model Design with Blocks Diagram.

performed in a repetitive and timed way to capture system events (which trigger various actions in the system).

8. Detailed System Design: Internal Blocks Diagram

SysML Internal Blocks diagram allows displaying of internal connections between the parts of a block. Through this diagram it is possible to define, directly or via ports, which parts of blocks are related and in communication with parts of other blocks. Ports are a special class of property used to specify allowable types of interactions between blocks. Figure 6 presents the individual characteristics of a communication block. Ports IPS_p4 and IPS_p3 are used to explain the interaction of assM (of Assembly_Module) and bus (Bus_Control). Connection between these ports is represented by associating connectors.

Another important contribution of a port is the definition of provided and required interfaces of the system in early phases of the development process. A provided interface is described by a circle and identifies a port that produces/provides services to its customers.

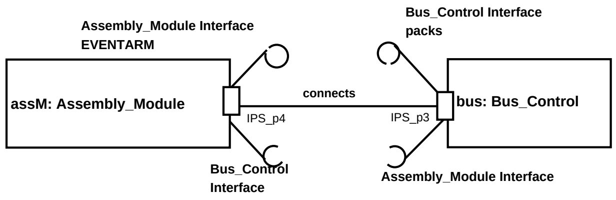

Figure 6. Model Design with Internal Blocks Diagram

9. Conclusion

This proposed research presented the application of an embedded real-time requirements modelling methodology for the development stages of CPS. SysML and MARTE compose a modelling proposal suitable to the representation of functional and non-functional requirements that allows to strengthen the phases of specification, analysis and design of systems. Moreover, VLS is applied in the proposed methodology to perform the requirement descriptions and annotations in a formal way, in modelling elements, contributing to decrease the ambiguity of specifications.

As contributions of this research, it can be highlighted the elaboration of a methodology able to represent not only software, but also the hardware components and their embedded controls. Furthermore, it can be also stressed the description of a fairly simple methodology for analysis, specification and modelling of complex systems at different levels of abstraction. Thus, system designers do not need to have knowledge of different modelling tools for the requirements specification process. Finally, the proposed methodology uses specific diagrams suitable to requirements and design description that are enriched with stereotypes of MARTE, a profile known for the purpose of modelling embedded real-time systems. In this context, the MBE approach allows improvement of communication between the hardware and software teams, once they use specifications of structure and behavior of the system at the same level of abstraction, using the same modeling languages.

Stereotypes of the MARTE profile are a convenient way of identifying elements in a UML model that have an additional non-standard semantics, that is, the semantics that goes beyond the limits of the UML standard in itself. Guidelines are presented in this paper for the adoption of an intuitive way for applying the MARTE profile MBE strategies for CPS design.

Combination of VSL, in formal specification models of requirements, with temporal annotations of MARTE, contributes to the decrease of ambiguity in specifications. Overall, this formalism will allow the analysis and verification of the correctness and consistency of artifacts created during specification process, through automated techniques, already in initial stages of development of these systems. Thus, specification errors may be found earlier through the development cycle.

Acknowledgment

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

References

[1] M. A. Wehrmeister, "An Aspect-Oriented Model-Driven Engineering Approach for Distributed Embedded Real-Time Systems," Thesis, Universidade Federal do Rio Grande do Sul, Porto Alegre, Rio Grande do Sul, 2009.

[2] L. Ollinger, M. A. Wehrmeister, C. P. Pereira, and D. Zuhlke, "An Integrated Concept for the Model-Driven Engineering of Distributed Automation Architectures on Embedded Systems," in *Proc. of the 11th IFAC Workshop on Intelligent Manufacturing Systems*, vol. 11, 2013.

[3] F. Hu, Y. Lu, A. V. Vasilakos, Q. Hao, R. Ma, Y. Patil, T. Zhang, Y. Lu, X. Li, and N. N. Xiong, "Robust CyberPhysical Systems: Concept, Models, and Implementation." *Future Generation Computer Systems*, vol. 56, pp. 449–475, 2016.

[4] E. Wozniak, M. Di Natale, H. Zeng, C. Mraidha, S. Tucci-Piergiovanni, and S. Gerard, "Assigning Time Budgets to Component Functions in the Design of Time-critical Automotive Systems." in *Proc. of the 29th ACM/IEEE International Conference on Automated Software Engineering*, 2014, pp. 235–246.

[5] S. Walter, A. Rettberg, and M. Kreutz, "Towards Formalized Model-Based Requirements for a Seamless Design Approach in Safety-Critical Systems Development." in *Int. Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORC)*, 2015, pp. 111–115.

[6] M. S. Soares and J. L. M. Vrancken, "A Modular Petri Net for Modeling and Scenario Analysis of a Network of Road Traffic Signals," *Control Engineering Practice*, vol. 20, no. 11, pp. 1183–1194, 2012.

[7] F. G. C. Ribeiro and M. S. Soares, "An Approach for Modeling Real-Time Requirements with SysML and MARTE Stereotypes," in *Proc. of the 15th Int. Conf. on Enterprise Information Systems*, 2013, pp. 70–81.

[8] M. A. Wehrmeister, E. P. Freitas, A. P. D. Binotto, and C. E. Pereira, "Combining Aspects and Object-Orientation in Model-Driven Engineering for Distributed Industrial Mechatronics Systems," *Mechatronics*, vol. 24, no. 7, pp. 844–865, 2014.

[9] H. G. C. Góngora, T. Gaudré, and S. Tucci-Piergiovanni, *Complex Systems Design & Management: Proceedings of the Third International Conference on Complex Systems Design & Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ch. Towards an Architectural Design Framework for Automotive Systems Development, pp. 241–258.

[10] G. Barbieri, C. Fantuzzi, and R. Borsari, "A Model-Based Design Methodology for the Development of Mechatronic Systems," *Mechatronics*, vol. 24, no. 7, pp. 833–843, 2014.

[11] R. Kraus, G. Papaioannou, and A. Sivan, "Application of Model Based System Engineering (MBSE) Principles to an Automotive Driveline Sub-System Architecture." Master Thesis, University of Detroit Mercy, Estados Unidos, EUA, 2016.

[12] B. Vogel-Heuser, D. Schutz, T. Frank, and C. Legat, "Model-Driven Engineering of Manufacturing Automation Software Projects A SysML-Based Approach," *Mechatronics*, vol. 24, no. 7, pp. 883–897, 2014.

[13] M. A. Wehrmeister and G. R. Berkenbrock, "AMoDE-RT: Advancing Model-Driven Engineering for Embedded Real-Time Systems." in *16th Int. Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, 2013, pp. 1–7.

[14] S. Hodges, A. Thorne, A. Garcia, J. Chirn, M. Harrison, and D. McFarlane, "Auto-ID Based Control Demonstration Phase 1: Pick and Place Packing with Conventional Control," Institute for Manufacturing, Tech. Rep. Auto-ID Centre Whitepaper CAM-AUTOID-WH-006, June 2002.

[15] J. Brusey, M. Fletcher, M. Harrison, A. Thorne, S. Hodges, and D. McFarlane, "Auto-ID Based Control Demonstration Phase 2: Pick and Place Packing with Holonic Control," Institute for Manufacturing, Tech. Rep., February 2003.

[16] A. Burns and A. J. Wellings, *Real-Time Systems and their Programming Languages: ADA 95, Real-Time Java, ad Real-Time POSIX.*, 3rd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., 2001.

[17] U. OMG, *OMG - OMG Systems Modeling Language - version 1.4.*, 2015, technical Report Formal/2015-06-03.