An Approach to Model-Driven Architecture applied to Hybrid Systems

Alessandro Gerlinger Romero [1] and Mauricio Gonçalves Vieira Ferreira[2]
National Institute for Space Research (INPE), São José dos Campos, São Paulo, 12227-010, Brazil

Hybrid systems are characterized by a composition of discrete and continuous dynamics. In particular, the system has a continuous evolution and occasional jumps. The jumps are caused either by controllable, uncontrollable external events or by its continuous evolution. The continuous evolution and these jumps in control loops are the origins from the most stringent real-time demands. With the necessity to launch more satellites, Brazilian National Institute for Space Research (INPE) has been carrying out research on modeling and verifying hybrid systems, of which its main focus is to obtain a better balance between dependability, schedule, and cost. We are attempting to use Object Management Group (OMG) specifications to model discrete events. We are focusing mainly in Modelica (with some degree of Scicoslab) to model continuous dynamics. Another concern addressed by this, INPE, research is to be independent from commercial tools, establishing itself on open source software. This paper presents an approach to implement Model-Driven Architecture in hybrid systems based on vendor neutral specifications. It shows how the models are defined, traced and used, as well as a set of tools for this. SysML (Systems Modeling Language), and MARTE (Modeling and Analysis of Real-Time Embedded Systems) allowed us to define a Computation Independent Model focused mainly on high-level structure and behavior (state oriented). At the end, a case study is presented (inverted pendulum). From this case study, we have concluded that the proposed approach can complement uncovered topics in current research applied to hybrid systems development and maintenance.

I. Introduction

HYBRID systems are in all kinds of devices from cars up to spaceships. They are systems characterized by continuous dynamics and occasional jumps in these dynamics. The jumps are caused either by controllable, uncontrollable external events or its continuous evolution. The continuous evolution and these jumps in control loops are the origins from the most stringent real-time demands. These real-time demands are one of the reasons that hybrid systems usually require a high level of safety.

Usually, they are dedicated to manage the execution of a process or object of the physical world. This type of system consists of three main activities: measuring - related to sensors; command synthesis - related to a processing unit; and information output - related to actuators. Frequently, actuators are transducers that convert electricity into mechanical energy. Indeed, it is a commonplace that a hybrid system is a mechatronic system.

According Thramboulidis (2010), the traditional development process is wholly inappropriate for the development of mechatronic systems. Shah *et al.* (2010) emphasize that considerable research has been done for the automation of software development through UML although little effort has been focused on the model-based engineering of entire systems, including both software and hardware. Nonetheless, Model-Driven Architecture (MDA) is not a new approach in software engineering even though it remains far from being commonly used on projects in real systems (Giese *et al.*, 2010).

Dickerson and Mavris (2009) explain that separation of problem specification from the solution is accomplished by the logical separation of the Computation Independent Model (CIM) and Platform Independent Model (PIM) in MDA. However, Dickerson and Mavris (2009) state that relationships between the concepts and terminology of

[1] Doctoral student of course Space Engineering and Technology-ETE, Option Engineering and Management of Space Systems-CSE at INPE, Av. dos Astronautas, 1758, building CCS, São José dos Campos, 12227-010, Brazil, romgerale@yahoo.com.br
[2] Doctor Engineer/Researcher, CRC (Satellite Tracking and Control Center), Av. dos Astronautas, 1758, building CCS, São José dos Campos, 12227-010, Brazil, mauricio@ccs.inpe.br

systems engineering and MDA have not been firmly established, much less agreed upon by the software and systems engineering communities.

In this paper, we present an approach to MDA (OMG, 2003) applied to hybrid systems. We explore two of the three models defined by MDA in systems engineering: CIM and PIM. In addition, we place emphasis on two different design domains typically involved in the development of hybrid systems. The first domain is the systems engineering approach towards the system design including discrete behavior. The second domain is the analysis of the continuous dynamic behavior, along with the control system design. Furthermore, a third domain is explored, the real-time properties required by the system.

For each domain addressed by this paper, there is a vendor neutral specification:

- SysML (OMG, 2010a) is used to define CIM and PIM;
- Modelica (Modelica Association, 2012) is used to simulate analytical models;
- MARTE (UML Profile for Modeling and Analysis of Real-Time Embedded Systems) (OMG, 2011) is used to collect real-time constraints in CIM and PIM.

SysML and Modelica are connected using draft version of SysML-Modelica Transformation Specification (OMG, 2010b).

The remainder of this paper is organized as follows. In the next section, related works are presented. Section III presents our initial approach that emphasizes what responsibilities are allocated to CIM and PIM. In Section IV, a case study based on inverted pendulum problem is presented and discussed. Finally, conclusions are presented in Section V.

II. Related Works

Cloutier (Cloutier, 2006) advocates use of MDA in system engineering. According this author, systems engineers begin modeling business rules and concept of operations (CONOPS) in a CIM. The systems engineer develops use cases which capture who are the users of the system, what are the uses and capabilities of the system, and the significant communications between the system and its environment. The systems engineer also begins to build the domain information model during this phase. CIM is transformed in PIM that has system requirements, system architecture and high level design.

Advanced Research and Technologies for Embedded Intelligence and Systems (ARTEMIS) (Obermaisser and Kopetz, 2009) is a coordinated European effort to establish foundations for embedded systems. MARTE and MDA have selected as basis for real-time modeling in ARTEMIS. These authors use the term conceptual modeling instead of CIM. According these authors, conceptual modeling is the phase where an abstract behavior and structure models of the system are designed in order to get understanding of the problem and transfer this understanding to the developers involved on the system architecture design. PIM is defined in three phases of architecture design: the application architecture design, platform architecture design and system architecture design, the last being the result of the System Allocation/Configuration/Refinement phase.

Qamar *et al.* (2011) use the term conceptual design that, according these authors, is a dynamic phase in terms of change in design and interaction of designers where initial product synthesis serves as a basis for developing an abstract function structure and corresponding function principles.

Buckl *et al.* (2010) focus on real-time embedded systems. They define and explore alternatives for specifying properties of time. This paper highlights that a modeling language should allow the description of the system temporal constraints regardless of a specific solution. Furthermore, Buckl *et al.* (2010) explore and compare the MARTE (OMG, 2011) with other alternatives for setting properties of time.

Regarding real-time software, Giese *et al.* (2010) state that the functionality is developed regardless of the platform and its interfaces with the environment (A/D converters and D/A). Therefore, such models ignore properties like WCET, hardware features, memory consumption or power. Focusing on the logical order of execution and data flow, this model allows verification and validation through simulation, using either no plant or a plant model as environment. The objective of this stage is to run a first proof of concept, verification and validation of the overall project and its control laws.

Project P (OpenDo, 2012) is a coordinated effort started in late 2011 to support the model-driven engineering of high-integrity embedded real-time systems. This project is working to provide a framework able to verify the semantic consistency of systems described using subsets of heterogeneous modeling languages, ranging from behavioral to architectural languages such as SysML, MARTE and Scicos (SCICOSLAB, 2012).

Hien and Quang (2012) describe CIM using a usecase model, a kind of functional diagram and hybrid automata to model autopilot systems of ships. Qamar *et al.* (2011) defines a mechatronic design infrastructure using

SysML to establish a domain-independent system model, thus establishing relationships and means for automated integration with other design models. Qamar *et al.* (2010) present options to identify and to represent systems dependencies inside a model, and to connect a system descriptive model with detailed-design tools. Thramboulidis (2010) proposes an integration of the mechanical, electronic and software systems though the 3+1 SysML view-model. SysML is used to specify the central view model of the mechatronic system while the other three views are for the different domains involved. Shaa *et al.* (2010) presents a framework in which multiple views can automatically be generated from a common system model defined using SysML.

Schamai *et al.* (2009) present a graphical Modeling Language (ModelicaML), a UML Profile for Modelica, which enables an integrated modeling and simulation of system requirements and design (for systems including both hardware and software). Sjöstedt *et al.* (2008) shows limitations with SysML parametric diagrams for modeling dynamic systems and possible ways to overcome this problem.

Finally, Uttamag (2009) models and simulates an inverted pendulum focusing on SysML features.

III. Our initial approach

According to the OMG (OMG, 2003), the goal of MDA is to provide an open, vendor neutral approach to the challenge of business and technology change. In MDA, a CIM is transformed into a PIM.

CIM focuses on the environment of the system and its requirements. The details of the structure and behavior are not specified. It plays an important role in reducing the gap between those that are experts about the domain and its requirements on one hand, and those that are experts of the design of the system that satisfy the domain requirements, on the other (OMG, 2003).

PIM presents a view of the system independent from the target platform and can be considered a functional description of the system. A PIM exhibits a specified degree of platform independence so as to be suitable for use with a number of different platforms of similar type.

Next subsections explore CIM and PIM modeling and simulation. The section IV presents a problem oriented explanation.

A. CIM

In our initial approach, conceptual modeling is performed through a CIM using SysML.

The term conceptual modeling was allocated from Obermaisser and Kopetz (2009) to CIM.

Systems engineers begin modeling a mission domain where the mission context establishes the communication between system, its environment, and its users. This communication is accomplished by physical connections (*Flowports*) and by interaction with users (*Standard ports*). Physical connections demand definition of mission *ValueTypes*.

Mission context analysis results in *Blocks* for entities in environment and one *Block* for the system. In case studies, one single *Block* is used for the system under definition in CIM. This is a straightforward alternative to avoid detailed specification in early phases; and, consequently, to choose an abstract solution while in CIM. According Dickerson and Mavris (2009), problem specification is performed so that the system can be visualized as a black-box. Mission structure is defined by *Blocks*.

Mission usecases describing uses and capabilities of the system are gathered and analyzed based on mission context and structure. Mission behavior emerges from usecases and is defined by modes in the system. Considering system as a black-box, these modes are defined by a state machine. Transitions between modes often can require real-time constraints. HLAM (High-Level Application Modeling) package from MARTE is used to model real-time constraints gathered in CIM.

Mission requirements and testcases are defined based on previous modeled elements. Eventually, a top level objective function is defined in CIM to drive evaluation of multiples abstract solutions described by PIMs.

Behind modeling with CIM, there is the following statement: focus on problem, use SysML to start from the context, analyze interfaces from “outside in” viewpoint, adhere to a top down approach, and as soon as they emerge gather non-functional properties such as real-time constraints.

B. PIM

In our initial approach, CIM is transformed into PIM manually. PIM is defined using SysML.

The term conceptual design was allocated from Qamar *et al.* (2011) to PIM.

PIM using SysML contains a complete picture of the system from a higher abstraction level viewpoint. PIM is the descriptive system model, defined by OMG (OMG, 2010b) as a model that is used to capture descriptions of a

systems or concepts generally in terms of their features and relationships. It captures multiple aspects of the system, including its functionality, inputs/outputs and control flows, structural compositions and interconnections, and traceability to its mission and system requirements. It is a common model for different domain experts to define and specify the system.

In this approach, each PIM is coupled with an abstract solution. For instance, section IV shows a PIM using an inverted pendulum as abstract solution to mission requirements defined in CIM *StabilizedRod* but others PIMs can be modeled and evaluated as: double inverted pendulum, and a rod fixed to a cart. For each PIM, systems requirements tend to be different. Furthermore, each PIM exhibits a degree of platform independence; e.g., the rod can have one meter or one centimeter, controller can be implemented using software or an ASIC (Application-Specific Integrated Circuit), control laws can be designed using classic or modern control, etc…

PIM refines structure through decomposition of CIM *Block* system. Each *Block* of system is defined but we avoid deep diving into internal details. Every environment and user communication is refined, and also communication between internal *Block*s of system. Also PIM refines behavior. It is common to evaluate required frequency for sensors and actuators according control design. These real-time constraints are modeled using MARTE – HRM (Hardware Resource Modeling).

All mission requirements must be *satisfied* by, at least, one element in PIM. System usecases and requirements are gathered and analyzed starting from structure and behavior in PIM and whole CIM. Testcases are derived from system requirements. At this point, we have system structure, system behavior, system usecases, system testcases, and system requirements.

System descriptive model can be defined considering discrete behavior simulation. More information about PIM modeling and simulation using OMG specifications can be found in Romero (2010).

However, system requirements analysis and design need many different engineering analyses to evaluate the extent to which the system can satisfy its system performance, physical, reliability, maintainability, and cost requirements (OMG, 2010b). Engineering analysis is applied to evaluate critical system parameters.

As it was stated by Shah *et al.* (2010), the development of embedded systems requires more than one single language, such as UML or SysML, to effectively capture all of the needed information. On the other hand, creating models at different abstraction levels is necessary to unleash the hidden complexity present inside the system which is otherwise not visible while modeling at a single abstraction level (Qamar *et al.*, 2010). Nevertheless, by adding abstraction levels, an integration issue emerges, where different tools are used to perform simulation and analysis. These tools have to explicitly capture the sub-system design requirements broken down from the mission and systems requirements.

A part of this integration issue can be addressed by SysML4Modelica transformation specification (OMG, 2010b). This specification defines stereotypes used to create an analytical representation from the system structure of the PIM, so this analytical model is transformed to Modelica programs, which can then be simulated. Results collected from simulation feedback system descriptive model.

Only *Blocks* and *Internal Block Diagrams* are used in the SysML4Modelica profile (OMG, 2010b) because complexity inherent on parametric diagrams can often be abstracted from the system descriptive model. This is in accordance with Qamar *et al.* (2010), which argue that in the long run, it is better to define only input and output parameters in SysML descriptive models. In this paper, equations are modeled in the analytical model according SySML4Modeliva profile.

Modelica can be combined with Scicoslab (SCICOSLAB, 2012) using scicos *Modelica Generic Block 2* (Naja and Nikoukhah, 2006). This gives to our initial approach a very flexible alternative to simulate analytical models defined in PIM.

In conclusion, our initial approach generates a PIM defined by system descriptive model using SysML. This system descriptive model has high level structure and discrete behavior, non-functional constraints (real-time constraints defined by MARTE in this paper), usecases, testcases, and requirements. Analytical models for continuous dynamic behavior, along with the control system design, are transformed from SysML to Modelica programs. Scicos models can use Modelica programs to enhance simulation flexibility. Analytical models help to improve our understanding of the system requirements.

It is undesirable to get into too much detail in first level PIM, choosing a concrete solution. The first level PIM should remain as much as possible platform independent. Another PIM can be refined to explore the next lower level of abstraction.

When modeling PIM we consider the following: focus on abstract solution, use SysML to refine CIM introducing hierarchy in a system descriptive model, analyze interfaces from “outside in” viewpoint even between internal components, and simulate analytical models to evolve system requirements using multi-view modeling.

IV. Case Study

A case study considering the generalized version of the inverted pendulum's problem was developed to evaluate our initial approach. Specifications and models discussed above were applied to a mission called "StabilizedRod". We have chosen *StabilizedRod* to concentrate on gathering and analyzing mission requirements, and not on a particular abstract solution. This has allowed us to practice this main concept behind MDA and our initial approach presented above.

Indeed, the abstract solution was defined as an inverted pendulum. The inverted pendulum is a model of the attitude control for satellite launch vehicles at its departure. The objective of the attitude control problem is to keep the vehicle in a vertical position. The uniqueness of the inverted pendulum, due to its natural instability, provides various research areas: systems, control, electronics, and software. Furthermore, the inverted pendulum is a classic hybrid system, since it is composed of continuous dynamics (stabilization of the pendulum in a vertical axis) and discrete logics (mode management).

We manipulated the classical inverted pendulum to put it in an industrial device. Case study started from one stakeholder requirement:

- An operator can use a device that must move a rod to right and to left.

A. CIM - StabilizedRod

As it was stated previously, CIM is a model, which shows the system in the environment where it will operate. Consequently, the CIM modeling starts representing mission context to *StabilizedRod*. Figure 1 shows a block definition diagram to mission context after iterations.

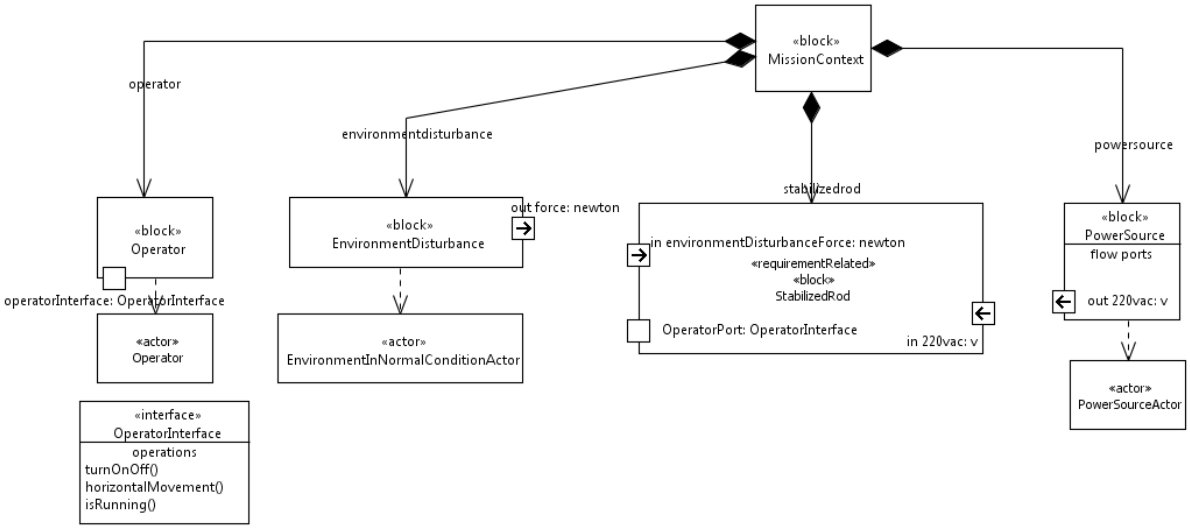


Figure 1. *StabilizedRod* mission context.

In our initial approach, mission context is part of domain modeling together with mission ValueTypes, mission structure and mission behavior. Mission ValueTypes is very important because it defines characteristics from flow ports. Figure 1 shows flow ports, and it ValueTypes. A standard port, defined by an interface called *OperatorInterface,* was modeled in mission behavior. This standard port represents how Operator actor gives and receives stimulus from *StabilizedRod*.

These elements were expressed in an internal block definition diagram showing its connections. This diagram highlighted that using standard port *OperatorInterface* an Operator interacted with *StabilizedRod*. These stimuli and how *StabilizedRod* responded was modeled using a state machine diagram showed in Fig. 2. This state machine defines *StabilizedRod* modes. Mode is one of the discrete parts of the system. All these modes were allocated to the *StabilizedRod* in CIM. At this level of abstraction, the *StabilizedRod* is the only element that represents the system.

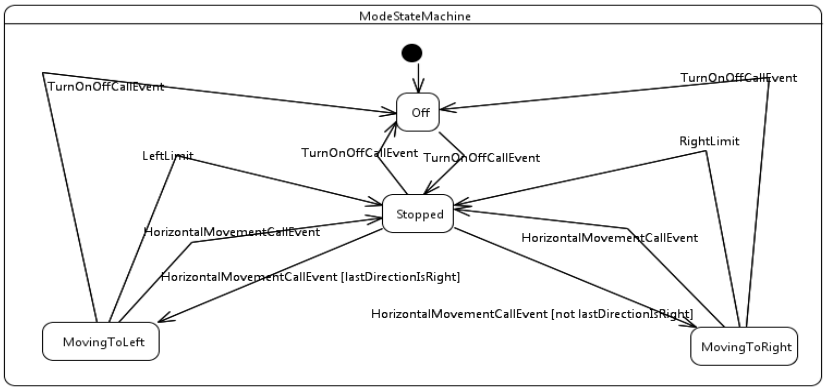


Figure 2. StabilizedRod modes.

Usecases were developed to analyze CONOPS and manufacturing; iteratively, comprehension about mission requirements became more and more clear.

During analysis of *Adjust horizontal position* mission usecase, it was gathered that when an operator requests to stop or move the *StabilizedRod*, it should respond in a maximum of 100 milliseconds. This non-functional property has modeled using MARTE HLAM. As show in Fig. 3, the operation *horizontalMovement* in *OperatorInterface* was stereotyped using *rtFeature* and its *rtSpecification* specifies arrival pattern as *aperiodic*, deadline as 100 milliseconds and as a hard deadline.

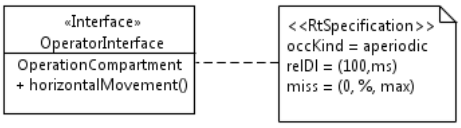


Figure 3. Real-time definition in CIM.

At this point, part of the requirements and related elements were able to be expressed using the requirements diagram showed in Fig. 4. A requirement is copied to operation package, and then modeled using decomposition and derivation. Usecases, actors and blocks were associated to it using refine and trace relationships.

A rationale related to *NormalCondition* requirement is defined; in fact, it is specified in detail to be used in analytical continuous models.

A testcase responsible to verify the requirements related to the rod's horizontal movement was depicted showing that mission test cases were defined.

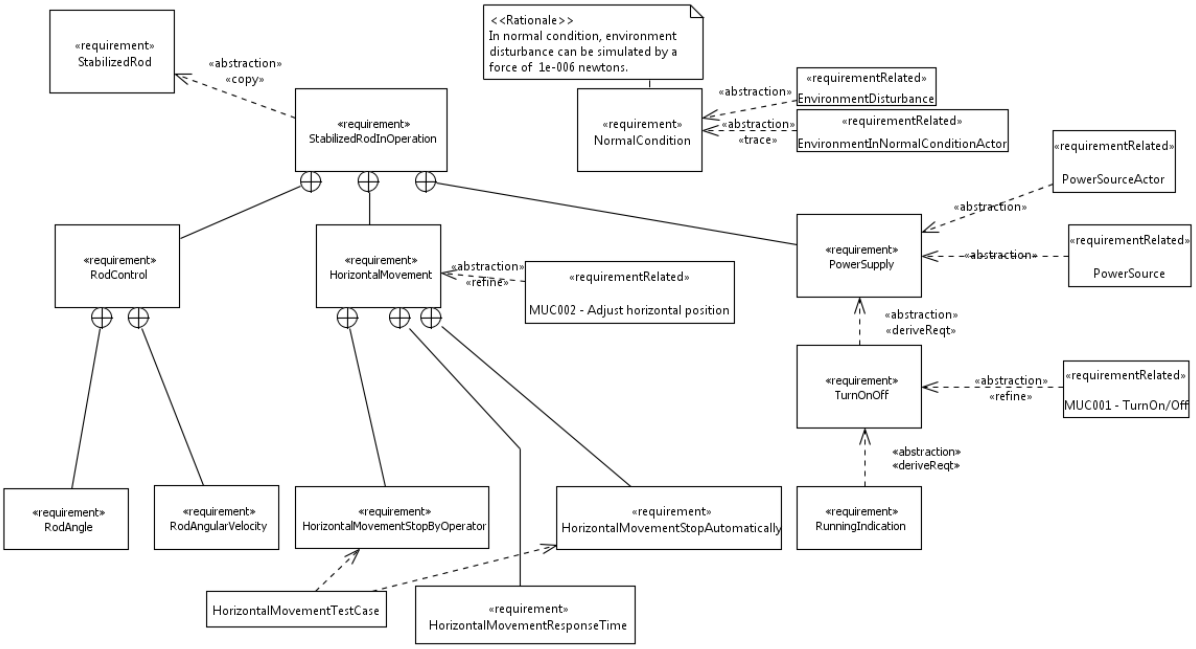


Figure 4. *StabilizedRod* mission requirements - operation.

The modeled CIM had views that conforms viewpoints. It was used to group concerns that were dispersed in model. CIM modeled had three viewpoints: CONOPS, Manufacturing, and Real-time. The CIM model had five main packages: Mission Requirements, Mission Usecases, Mission Domain, Mission Verification and Views.

Using the above constructions all mission requirements are gathered, analyzed and recorded in a model. This model did not define a particular abstract solution. This allows evaluation of a variety of abstract solutions that satisfies these mission requirements.

B. PIM – *StabilizedRod* as an Inverted Pendulum

In this paper, the selected abstract solution to mission requirements is an inverted pendulum but a rod fixed to a cart or a double inverted pendulum are valid abstract solutions. Multiples abstract solutions can also be described, each one by one PIM, and evaluated to yield the option that offers better results on top level objective function defined in CIM. In this paper, inverted pendulum was chosen to enable engineering analysis using a well known problem.

The transformation from CIM to PIM was performed manually in this paper.

Now that CIM has defined mission requirements and domain, PIM modeling begins gathering and analyzing system structure, considering the mission context defined in CIM. All definitions in PIM must be a refinement from the mission modeling, e.g. the *StabilizedRod* must have two input flow ports defined in mission context of CIM.

Figure 5 shows the first block definition diagram for system structure in PIM.

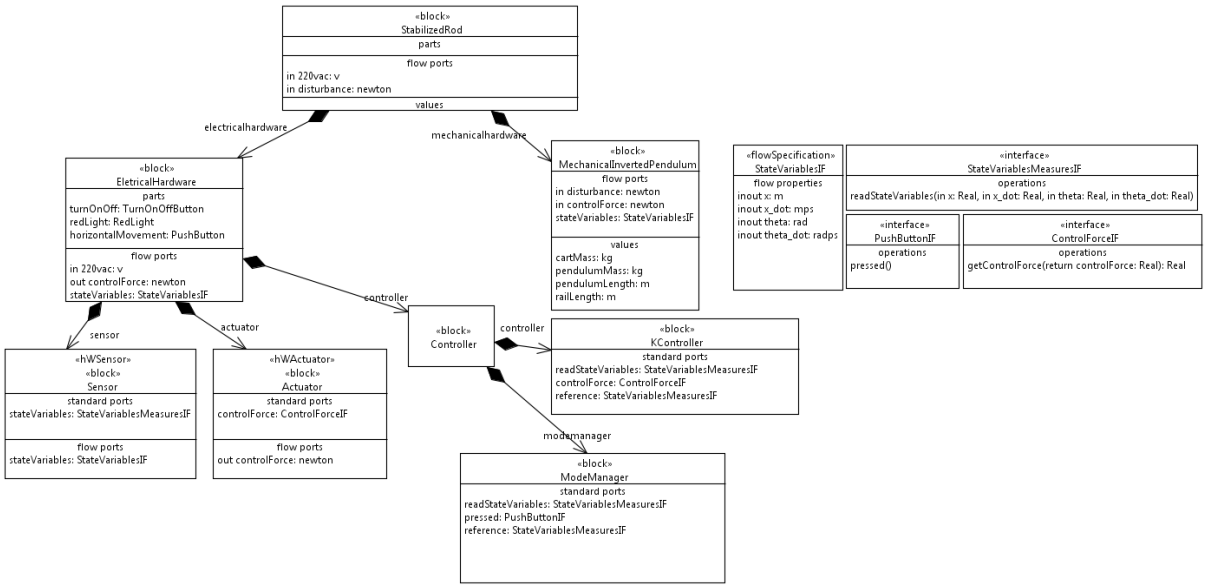


Figure 5. *StabilizedRod* structure.

As shown in Fig. 5, system structure was decomposed using hierarchy.

The mechanical domain was represented by *MechanicalInvertedPendulum*, and the electrical domain is represented by *ElectricalHardware*. *ElectricalHardware* has two special blocks: (1) *Sensor* stereotyped with MARTE HRM *hWSensor;* (2) *Actuator* stereotyped with MARTE HRM *hWActuator*. These blocks are responsible for transforming a mechanical movement into an electrical signal, or vice-versa. Each block has an A/D converter or a D/A converter because it is assumed that the *Controller*, and its components, works with digital signals. Furthermore, *ElectricalHardware* transforms an electrical signal exchanged by flow ports to standard ports. In this PIM, the Controller can be implemented with multiple alternatives such as: ASIC, FPGA (Field-Programmable Gate Array), microcontrollers or digital computer.

Interfaces were defined using SysML *Interface* for standard ports and SysML *FlowSpecification* to flow ports, which are not based in atomic flow ports.

MechanicalInvertedPendulum shows four value properties that have no default value, these parameters are subject of engineering analysis. Others value properties were also defined but they were not relevant for this paper.

Figure 6 presents the internal block definition diagram for the *StabilizedRod*. It is possible to notice that the *Sensor* and the *Actuator* allow communication from the *Controller* (based on digital signals) to mechanical

components. This reflects the "*outside in*" approach, as mentioned earlier, where interfaces to other components are modeled first.

The internal block definition for the *Controller* defines how it uses signals collected from *Sensor* to send to the *ModeManager*, and, eventually, getting signals generated by the *ModeManager* redirected to the *Actuator*. *ModeManager* was supplier from *allocation* of a majority of modes defined in CIM (from Fig. 2 modes: Stopped, MovingToRight and MovingToLeft), and refined in PIM. It defines reference values to *KControler* based on current state variables and mode.

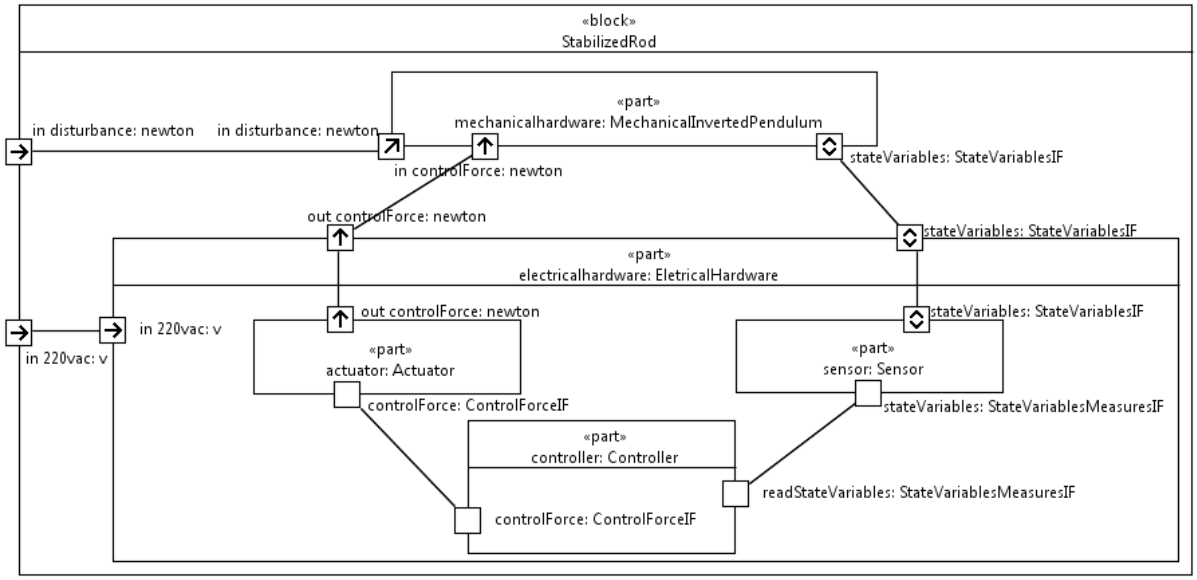


Figure 6. *StabilizedRod* internal block definition diagram.

System usecases were defined considering system functions. Iteratively, as PIM was modeled understanding about system requirements became clearer. Table 1 shows part of system requirements. Prefixes used in requirement id are to indicate level: (MR) Mission Requirement; (SR) System Requirement. Mission requirements were *copied* from CIM and decomposed based on the selected abstract solution. System requirements were grouped in PIM using two categories: product system requirements, and process system requirements.

Table 1. *StabilizedRod* part of system requirements.

	[Label]	text	id
1	StabilizedRod	The system shall provide a stabilized rod.	MR001
2	StabilizedRodAsInvertedPendu	The system shall provide a stabilized rod using an inverted pendulum.	SR101
3	RodControl	The system shall control angle and angular velocity from rod in normal condition.	MR002
4	RodControlInvertedPendulum	The system shall control angle and angular velocity from rod by move the cart horizontally in normal condition.	SR102
5	HorizontalMovement	The system shall move the rod on horizontal axis to left and right from an operator command in normal condit...	MR003
6	HorizontalMovementPushButt	The system shall move the rod on horizontal axis to left and right by operator command received from a push bu	SR103
7	RunningIndication	The system shall indicate that it is running.	MR009
8	RedLight	The system shall emit a red light when it is turned on.	SR104

Furthermore, starting from system structure and behavior, engineering analysis is applied to evaluate critical system parameters. Engineering analysis is a critical aspect of systems engineering.

As it was stated in our initial approach, it was defined blocks in analytical model. These blocks were stereotyped with SysML4Modelica profile. Therefore, system descriptive blocks were allocated to analytical blocks (stereotyped with SysML4Modelica). These stereotyped blocks were transformed in Modelica programs then they can be run in a Modelica environment for simulation purpose. These simulations generate data that is used to refine system structure, behavior or parameters, and, eventually, system requirements.

Figure 7.a. shows blocks defined to evaluate inverted pendulum dynamics using parameters, controller, and influence caused by environment according to the requirement and rationale stated in CIM. Each block stereotyped with *ModelicaBlock* or *ModelicaModel* generates one Modelica program that can be run in a Modelica environment.

In addition, *InvertedPendulumBlock* and *ControlledInvertedPendulum* defined simulation parameters. ValueTypes defined in the mission domain are used by value properties from blocks in analytical model; moreover, each one was stereotyped with *ModelicaValueProperty* indicating that it was a parameter. Each flow port was stereotyped with *ModelicaPort* defining its causality. Inverted pendulum dynamics evaluation was performed without use of Modelica Standard Library. Therefore, every equation was defined in each block with a constraints stereotyped with *ModelicaEquation*.

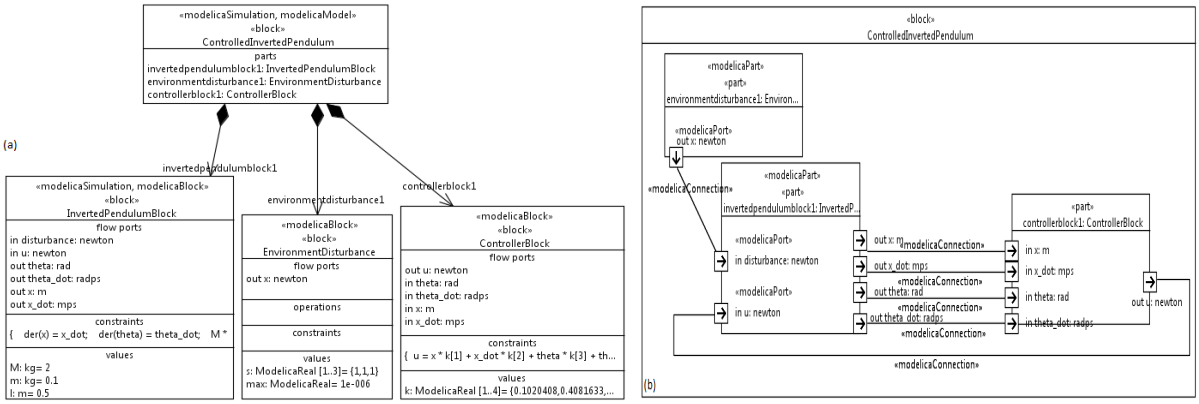
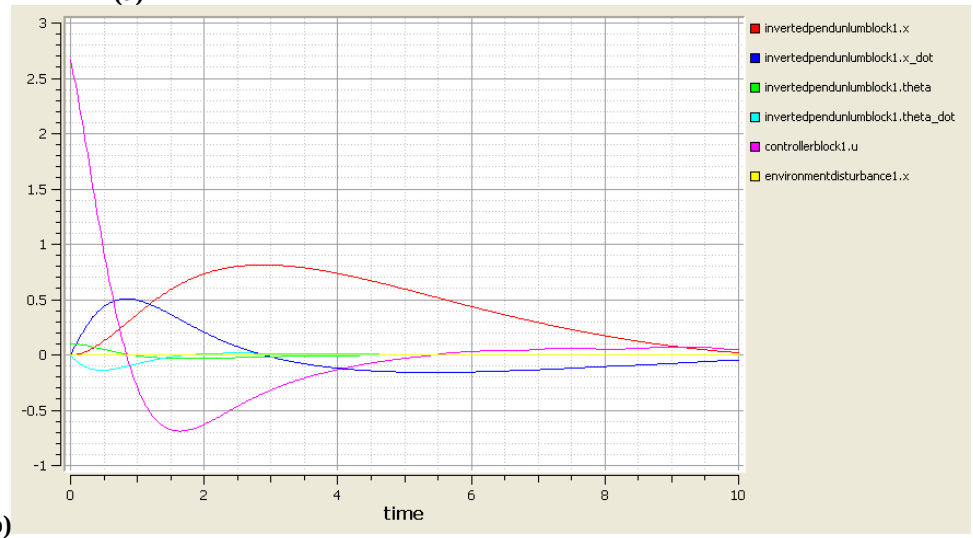


Figure 7. (a) Block definition diagram for analytical elements using SysML4Modelica; (b) Internal block definition diagram for these elements.

Figure 7.b shows internal block definition diagram for *ControlledInvertedPendulum.* These classes were transformed into Modelica programs; as a result, Fig. 8.a shows generated Modelica program for *ControlledInvertedPendulum*. A series of simulations was performed using these Modelica programs. Figure 8.b shows one simulation result for *ControlledInvertedPendulum* using initial theta as 0.1 rad. *ModeManager* could be modeled using Modelica Standard Library StateGraph but we chose to use Scicoslab to prototype this block.

```
model ControlledInvertedPendulum
  PendulumPackageSpaceOps.InvertedPendunlumBlock invertedpendunlumblock1;
  PendulumPackageSpaceOps.ControllerBlock controllerblock1;
  PendulumPackageSpaceOps.EnvironmentDisturbance environmentdisturbance1;
equation
  connect(invertedpendunlumblock1.u,controllerblock1.u);
  connect(invertedpendunlumblock1.disturbance,environmentdisturbance1.x);
  connect(invertedpendunlumblock1.theta_dot,controllerblock1.theta_dot);
  connect(invertedpendunlumblock1.theta,controllerblock1.theta);
  connect(invertedpendunlumblock1.x_dot,controllerblock1.x_dot);
  connect(invertedpendunlumblock1.x,controllerblock1.x);
end ControlledInvertedPendulum;
```

(a)



(b)

Figure 8. (a) Modelica program for *ControledInvertedPendulum* (b) Simulation result for *ControlledInvertedPendulum*.

A similar process was performed to generate a scicos analytical model but without a specific profile like SysML4Modelica. Scicos was used to prototype *ModeManager* using TCL/TK; in addition, this prototype had *horizontalMovement* push button so it was possible to simulate system response to mode's change triggered by an *Operator* still in PIM. *InvertedPendulumBlock* defined using Modelica was reused with little modification to integrate into scicos model. Figure 9.a shows scicos model where: *RandomGenerator* scicos block replaced Modelica program *EnvironmentDisturbance*; and, one constant and one *MATMUL* scicos block replaced Modelica program *ControllerBlock*. Figure 9.b shows a simulation result for the scicos model. The button labeled as "*ChangeMode*" simulates *PushButton*.

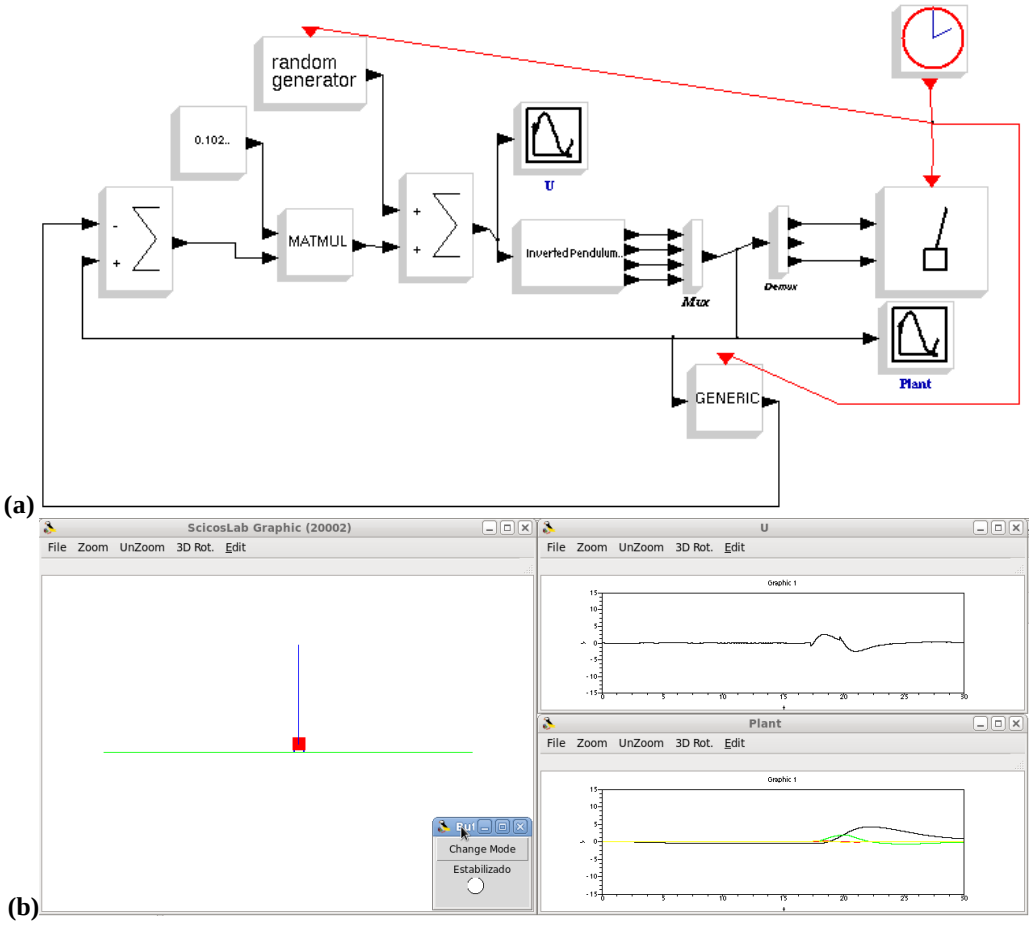


Figure 9. (a) Scicos model for *ScicosControlledInvertedPendulum;* (b) Simulation result for *ScicosControlledInvertedPendulum*.

These analytical blocks where allocated from descriptive model according Table 2.

Table 2. Allocation from descriptive elements to analytical elements.

	/eContainer	supplier	client
1	modelica	EnvironmentDisturbance	disturbance
2	modelica	InvertedPendulumBlock	MechanicalInvertedPendulum
3	modelica	ControllerBlock	KController
4	modelica	M	cartMass
5	modelica	m	pendulumMass
6	modelica	l	pendulumLength
7	scicos	ControlledInvertedPendulum	StabilizedRod
8	scicos	ModeManager	ModeManager, PushButton
9	scicos	ControlledInvertedPendulum	Controller

C. Discussion

The case study shows us that MDA and system engineering can be conciliated in an elegant way through SysML.

We concluded that CIM encompasses the term conceptual modeling that is sometimes called requirements analysis, in context of hybrid systems. In fact, CIM has an abstract behavior and structure that are designed in order to get understanding of the problem (Obermaisser and Kopetz, 2009). Dickerson and Mavris (2009) complement that problem specification is accomplished in CIM. Therefore, CIM is focused in problem space, and it is generated by mission requirements gathering and analysis process. In our case studies, we observed that this strong separation from problem space and abstract solution space is very valuable, as it can help to avoid mistakes that often occur when we go to abstract or concrete solution quickly.

We concluded that PIM encompasses the term conceptual design, in context of hybrid systems. In fact, Qamar *et al.* 2011 states conceptual design as a dynamic phase in terms of change in design and interaction of designers, where initial product synthesis serves as a basis for developing an abstract function structure and corresponding function principles. The first level PIM has the initial system architecture and it is refined iteratively henceforth. As it was stated in the case study, PIM represents an abstract solution. We stated some didactical examples about different abstract solutions in previous sections but we have used this rule to detect that a separated abstract solution is needed: when to accommodate a new abstract solution is required changes in system descriptive model.

The first level PIM is constrained by CIM, and constraint next lower level of abstraction. This PIM can be derived in others domains not explored in this paper using techniques proposed in literature (Shah *et al.*, 2010) (Qamar *et al.*, 2011).

In the case study, we illustrated a simple use of MARTE in CIM but this use can be extrapolated to other non-functional properties in CIM and PIM. In addition, other metrics can be described and evaluated; for instance, CONOPS soft real-time deadlines derived from system requirements such as: repair time described as a normal distribution with mean two days and standard deviation one day.

Our initial approach is based on vendor neutral specification, so it can be used with multiples tools. The presented case study is defined using these tools:

- Topcased 5.2 (TOPCASED, 2012) and Papyrus 0.8.2 (CEA, 2012) to develop all models;
- OpenModelica 1.8.1 (OPENMODELICA, 2012) to simulate Modelica programs;
- Scicoslab 4.4.1 (SCICOSLAB, 2012) to define and simulate scicos models.

Finally, we briefly compare the presented case study with two others in literature. We emphasized items where our initial approach can complement previous researches:

- Uttamang (2009) prefers to focus on SysML features. Nonetheless, he does not emphasize the separation from problem and solution space. At the same level, requirements were gathered, analyzed and documented, structure was defined in an intermediary level, and, some parts of structure were allocated to software.
- Qamar *et al.* (2011) starts from conceptual design without a previous conceptual modeling. Therefore, traceability and reuse of conceptual domain could be lost.

V. Conclusion

With the necessity to launch more satellites, Brazilian National Institute for Space Research (INPE) have been carrying research about modeling and verifying hybrid systems, the main focus is to obtain a better balance between dependability, schedule, and cost. As it was stated by Cloutier (2006), MDA is a key initiative to reduce cost while it can increase dependability.

As it was stated before, we are attempting to be vendor independent using vendor neutral specifications and open source tools. Therefore, this paper presented an approach to implement Model-Driven Architecture in hybrid systems based on vendor neutral specifications. It shows how models are defined, traced and used, as well as a set of open source tools for this.

Our initial approach contributes to conciliate system engineering and MDA covering topics not addressed by previous known researches. Also it presents a feasible complement to recent researches (Thramboulidis, 2010) (Shah *et al.*, 2010) (Qamar et al., 2010) (Qamar et al., 2011) on mechatronic systems, a special type of hybrid system. However, it is important to highlight that the current proposal can be applied towards hybrid systems. These are first results from our work so we do not intent to prescribe a process or even a method. We believe that the major contribution is related with CIM scope and use definition.

Future work targets: exploring CIM modeling, refining approach with discrete simulation in CIM and PIM concurrently to continuous simulation; refining non-functional properties analysis and simulation; SysML testcase

generation based on CIM; evaluation of automatic transformations using OMG MOF QVT or Triple Graph Grammars (TGG); evaluation about how patterns can be useful in refinements of our initial approach.

References

Buckl, C.; Gaponova, I.; Geisenger, M.; Knoll, A.; Lee, E. A. (2010). Model-Based Specification of Timing Requirements. In Proceedings… EMSOFT 2010 Proceedings of the tenth ACM international conference on Embedded software.

CEA (2012). Papyrus site. Available at: <http://papyrusuml.org>. Accessed on: 23 April 2012.

Cloutier, R. (2006). MDA for systems engineering – why should we care? USA: 2006. 10 p. Available at: <http://www.calimar.com/Papers/Model%20Driven%20Architecture%20for%20SE-Why%20Care.pdf>. Accessed on: 17 April 2012.

Dickerson, C.E.; Mavris, D.N. (2009). Architecture and Principles of Systems Engineering (CRC Complex and Enterprise Systems Engineering). Auerbach Publications. ISBN 978-1-420-07253-2

Giese, H.; Karsai, G.; Lee, E.A.; Rumpe, B.; Schätz, B. (2010). Model-Based Engineering of Embedded Real-Time Systems. International Dagstuhl Workshop, Dagstuhl Castle, Germany, November 4-9, 2007. ISBN 978-3-642-16276-3

Hien, N. V.; Quang, V. D. (2012). A realization model to develop the autopilot system of ships by specializing MDA. Vietnam Journal of Mechanics, VAST, Vol. 34, No. 1 (2012), pp. 55 – 65

MODELICA Association. (2012). Modelica Language Specification: Version 3.2R1. Available at: <https://www.modelica.org/news_items/documents/ModelicaSpec32Revision1.pdf> Accessed on: 17 April 2012.

Naja, M.; Nikoukhah, R. (2006). Modeling and simulation of differential equations in Scicos. In proceedings…The Modelica Association Modelica 2006, September 4th – 5th

Obermaisser, R.; Kopetz, H. (2009). Genesys – A candidate for an ARTEMIS Cross-Domain Reference Architecture for Embedded Systems. 2009. Available at: <http://www.genesys-platform.eu/genesys_book.pdf> Accessed on: 17 May 2011.

OBJECT MANAGEMENT GROUP (OMG) (2003). Model-Driven Architecture. USA: OMG, 2003. 62 p. Available at: <http://www.omg.org/mda>. Accessed on: 17 May 2011.

OBJECT MANAGEMENT GROUP (OMG) (2010a). Systems Modeling Language SysML: Version 1.2. USA: OMG, 2010. 260 p. Available at: < http://www.omg.org/spec/SysML/1.2/PDF >. Accessed on: 17 April 2012.

OBJECT MANAGEMENT GROUP (OMG) (2010b). SysML-Modelica Transformation Specification Draft. USA: OMG, 2010. 101 p. Available at: <http://www.omgwiki.org/OMGSysML/lib/exe/fetch.php?id=sysml-modelica%3Asysml_and_modelica_integration&cache=cache&media=sysml-modelica:sysml-modelica_xformspec_v.1.0_2010-5-10.pdf>. Accessed on: 17 April 2012.

OBJECT MANAGEMENT GROUP (OMG) (2011). UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems: Version 1.1. USA: OMG, 2011. 754 p. Available at: <http://www.omgmarte.org/>. Accessed on: 17 April 2012.

OpenDo (2012). Project P. Available at: <http://www.open-do.org/projects/p/>. Accessed on: 23 April 2012.

OPENMODELICA (2012). OPENMODELICA Site. Available at: <http://www.openmodelica.org/>. Accessed on: 01 April 2012.

Qamar, A., Törngren, M., During, C., Wikander, J. (2010). Integrating multi-domain models for the design and development of mechatronic systems. In proceedings… 7th Eusec Conference, may 2010. INCOSE, INCOSE.

Qamar, A.; Wikander, J.; During, C. (2011) Designing mechatronic systems: a model integration approach. In proceedings… International conference on engineering design, ICED11, August, 2011. Technical University of Denmark.

Romero, A. G. (2010). An approach to model-driven architecture applied to space embedded real-time software. Version: 2010-12-13. 203 p. Master dissertation (Space Engineering and Technology-ETE, Option Engineering and Management of Space Systems-CSE) - National Institute for Space Research (INPE), São José dos Campos, 2010.

Schamai, W.; Fritzson, P.; Paredis, c.; Pop, A. (2009). Towards Unified System Modeling and Simulation with ModelicaML: Modeling of Executable Behavior Using Graphical Notations. Proceedings of the 7th International Modelica Conference, Como, Italy. September 20-22, 2009.

SCICOSLAB (2012). SCICOSLAB Site. Available at: <http://www.scicoslab.org/>. Accessed on: 01 April 2012.

Shah A. A.; Kerzhner A. A.; Shaefer D.; Paredis C. J. J. (2010). Multi-View Modeling to Support Embedded Systems Engineering in SysML. Lecture Notes in Computer Science, Graph Transformations and Model-Driven Engineering, 2010, Volume 5765/2010, pp 580-601.

Sjöstedt, C. J.; Chen, D. J.; Cuenot, P.; Frey, P.; Johansson, R.; Lönn, H.; Servat, D.; Törngren, M. (2008). Developing Dependable Automotive Embedded Systems Using the EAST ADL; Representing Continuous Time Systems in SysML. In Proceedings of the 1st International Workshop on Equation-Based Object-Oriented Languages and Tools. Berlin, Germany, 2008.

Thramboulidis, K. (2010). The 3 + 1 SysML View-Model in Model Integrated Mechatronics. Journal of Software Engineering and Applications, Vol. 3, No. 2, 2010, pp. 109-118.

TOPCASED (2012). TOPCASED Site. Available at: <http://www.topcased.org>. Accessed on: 01 April 2012.

Uttamang, K. (2009). Design of Inverted pendulum System using SysML. No Magic Inc. Available at: < http://training.nomagic.com/index.php?option=com_docman&task=cat_view&gid=114&limit=20&limitstart=0&order=hits&dir=DESC&Itemid=72> Accessed on: 21 April 2012.