AIAA 2008-3309

A Software Modeling Approach Based on MDA for the Brazilian Satellite Launcher

Burgareli, L. A.[*]
Aeronautics and Space Institute - IAE, São José dos Campos, SP, Brazil, 12228-904
São Paulo University –EPUSP, São Paulo, SP, Brazil, 05508-900

Melnikoff, S. S. S.[†]
São Paulo University – USP, São Paulo, SP, Brazil, 05508-900

Ferreira, M. G. V.[‡]
Institute for Space Research – INPE, São José dos Campos, SP, Brazil, 12245-970

The purpose of this work is to define a strategy for the generation of Platform Independent Model (PIM) for the Brazilian Satellite Launcher (BSL) software, based on the Model Driven Architecture (MDA) approach. To accomplish that, a generic software model is created, using design patterns, to represent a family of satellite launcher vehicles whose characteristics are similar. From this generic model, PIMs are generated for variations of launcher vehicles, based on MDA approach, using executable UML (xUML). This strategy can contribute to the development of the software for same-family satellite launcher vehicles, as it defines a systematics for activities related to software modeling, and also allows the increase of BSL software models reuse and starts verification activity during the modeling phase. The BSL is a conventional launcher vehicle whose function is to launch to low altitude orbits, between 200 and 1000km, data collecting and remote sensing satellites, weighing between 100 and 200kg. As in the case of the BSL, space vehicle projects are usually long-term and, therefore, the replacement of obsolete technologies by more recent ones during development is likely. Also, new versions of a satellite launcher vehicle may use the software system in more modern computational platforms (hardware, operational system, language). Such replacements may lead to partial losses (models, architecture and code) or total loss of the software system, even before development is complete. This makes necessary a software development process which enables the reuse of the BSL software, regardless of changes in computational platforms and technologies. MDA is an approach of Object Management Group (OMG) which separates the system functionality specification from its implementation specification in a specific platform technology. It is a technique which, in addition to targeting reuse, places models as key factors in the software development. For the models to be reused, they must have been duly verified. The MDA approach is consistent with the idea of starting verification already in the modeling phase, through the concept of model content characterization, it can favor the verification activity. Also, the fact that the MDA approach may be based on xUML enables the start of verification during the modeling phase. Thus, this work presents a strategy to facilitate the reuse of BSL software models. A modeling approach is applied to the BSL software, using concepts of platform separation coming from MDA. A generic PIM is generated through a set of design patterns and distinct PIMs can be generated for each launcher variety using xUML. In this work, the need for a new modeling approach for the BSL software is justified. The MDA is succinctly presented, as well as MDA and critical systems related works. This work also describes the importance of starting a software verification since the modeling. The xUML and its relation to MDA and the verification of software models are also presented.

[*] MSc. – IAE/CTA, Av. Mal. Eduardo Gomes, 50, São José dos Campos, SP, Brazil /luciana@iae.cta.br.
† PhD. – EPUSP/USP, Av. Prof. Luciano Gualberto, trav.3, 158, São Paulo,SP,Brazil /selma.melnikoff@poli.usp.br.
[‡] PhD. – INPE, Av. dos Astronautas, 1758 - São José dos Campos, SP, Brazil / mauricio@iae.cta.br.

I. Introduction

The BSL is a conventional launcher vehicle whose function is to launch to low altitude orbits, between 200 and 1000km, data collecting and remote sensing satellites, weighing between 100 and 200kg [2]. As in the case of the BSL, space vehicle projects are usually long-term and, therefore, the replacement of obsolete technologies by more recent ones during development is likely.

Also, new versions of a satellite launcher vehicle may use the software system in more modern computational platforms (hardware, operational system, language). Such replacements may lead to partial losses (models, architecture and code) or total loss of the software system, even before development is complete. This makes necessary a software development process which enables the reuse of the BSL software, regardless of changes in computational platforms and technologies.

MDA is an approach of Object Management Group (OMG) which separates the system functionality specification from its implementation specification in a specific platform technology [16]. It is a technique which, in addition to targeting reuse, places models as key factors in the software development.

For the models to be reused, they must have been duly verified. The MDA approach is consistent with the idea of starting verification already in the modeling phase, through the concept of model content characterization, it can favor the verification activity [13]. Also, the fact that the MDA approach may be based on xUML enables the start of verification during the modeling phase.

Thus, this work presents a strategy to facilitate the reuse of BSL software models. A modeling approach is applied to the BSL software, using concepts of platform separation coming from MDA. A generic PIM is generated through a set of design patterns and distinct PIMs can be generated for each launcher variety using xUML. In section 2 of this work the need for a new modeling approach for the BSL software is justified. In section 3 the MDA is succinctly presented, as well as MDA and critical systems related works. Section 4 describes the importance of starting a software verification since the modeling. Section 5 defines xUML, its relation to MDA and the verification of software models. Section 6 shows an example of usage of the proposed strategy, applying it to a BSL telemetry module. Section 7 presents final considerations.

II. Motivation

In nearly 15 years of BSL software development, the development team has seen a high rate of qualified personnel rotation. Unfortunately, few developers have remained in the project since its inception. Therefore, it is necessary to ensure the understanding of clear documentation and updated models for the launcher, so that there are no difficulties in passing project information to new team members and all people involved can understand the intentions of original developers.

In addition to that, problems are also caused by the obsolescence of technologies used, such as methodology and CASE tools. The BSL software was originally designed using the technologies available in the early 90's, such as Structured Analysis using TeamWork, a graphical software-modeling tool originally marketed by Cadre. Today, it is ever more difficult to keep the infrastructure for such environments, making it difficult to capture, edit and update models and documents they generated. Therefore, a new development approach, which uses more recent technologies, improves understanding and helps the team to use and update models, is required.

Another important point is that during the long development process, hardware, operational system, compilers and sensors platforms underwent upgrades to new technologies, a trend in any project. Such changes may lead to partial losses (models, architecture and code) or total loss of the software system. Today, to keep up with BSL development it is necessary to develop or adapt software models specific for each launcher version, and that is a high-cost task. That evidences the need for the software development which allows increased reuse of BSL software models, preventing redundant efforts in the development of new systems, even in face of changes in computational platforms and technologies.

The perspective of new projects that provide for a gradual development of launchers, such as the one announced in 2005 by the Brazilian General Command for Aerospace Technology (CTA) and the Brazilian Space Agency (AEB) [2] [15], also creates the need for a more planned development of software, with the generation of models which are independent of platform so that they can be reused by the new satellite launcher vehicles.

Updating of models and documentation, replacing of obsolete technologies and perspectives for new projects are the motivation behind the presentation of a strategy to reuse BSL software models. A modeling strategy based on

MDA, providing for the reuse of software models, may solve a large part of current BSL software problems, and help in the development of new satellite launcher versions.

III. MDA in Critical Systems

MDA is an approach for system development based mostly on the importance of models [16]. The approach consists of the separation between Platform Independent Model (PIM) and Platform Specific Model (PSM). With this architecture separation proposal, the MDA approach offers portability, inter-operability and reuse of models [17].

In spite of being a recent concept, the MDA approach is being applied to more complex systems, such as embedded and real time systems. Benefits like cost reduction, flexibility and reuse are the main advantages encouraging developers to use MDA approach concepts in real time systems. Some works using the MDA approach in similar systems may be observed in: [4], [12], [19], [9] and [14].

IV. Software Model Verification

This section emphasizes the importance of starting the verification in the beginning of the software development, since the modeling phase. In the last decades there has been significant growth in the use of software in space systems, and that made adequate verification to such systems more difficult and more critical. Many times, the verification is made only at code level and not at model level [11]. That happens because many developers believe investments in verification efforts not to be worthwhile in phases prior to implementation, as verification will be repeated at code level. However, many times mistakes may be happening since the beginning of development and practical verification techniques must be applied since the modeling phase, so that faults do not persist from project phase to implementation phase.

The MDA approach is consistent with the idea of starting the verification in the modeling phase, as its architectural separation concept favors the verification activity, as it prevents testers to prematurely involve in implementation details, allows verification to be performed in defined phases of the modeling phase, and also allows that tests created for PIM, in case of migration, may be reused safely, as the PIM has no specification details. Some authors began to care for verification during modeling, and suggest verification in the MDA context, as works [13], [10], [7], [11], [5] and [8] make clear. The authors mention that many research works into MDA are ongoing, but challenges remain in terms of software test and verification in such contexts.

V. Executable Models - xUML

xUML is a UML profile which supports the execution of UML models, based on precise semantic actions [3] [10]. The constructive core of xUML includes class models, Finite State Machines (FSMs) and communication diagrams. An executable model is a set of interacting FSMs, and the execution of such model is carried out through simulating its behavior. When there is a transition between states, a procedure associated to the state is carried out. This procedure is a set of statements of an Action Semantic Language (ASL). ASL is an implementation independent language for specifying processing within the context of an xUML model.

The MDA approach relies on xUML, as a PIM may be expressed in xUML. Using the idea of execution, the PIM may be simulated, enabling the verification of its behavior. Therefore, using the MDA approach based on xUML may be a way of ensuring the verification begins in the modeling phase.

VI. PIM Generation Strategy

The strategy used aims at improving the reuse of BSL software models and encompasses launcher versions having similar characteristics. This strategy is based on the generation of a generic PIM, which is a generalization of models corresponding to a set of launchers with similar characteristics. The function of the generic PIM is to help in the generation of new PIMs, specific for other launchers. Figure 1 shows the idea of inclusion of a generic PIM in the MDA approach, compared to the traditional approach. The generic PIM has a higher level of abstraction. Figure 1 shows that instead of having a single PIM for a specific BSL and, from there create a number of PSMs, it is possible to have a generic PIM, encompassing BSLs variations and, from there, create a number of PIMs, each one corresponding to a given BSL.

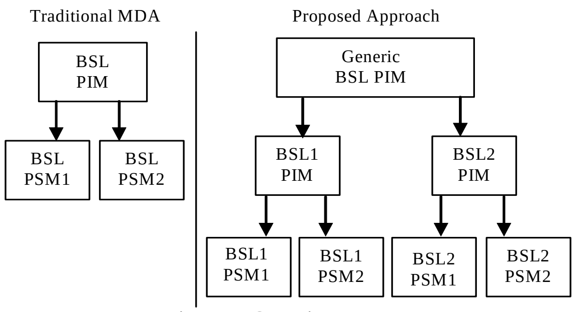

Figure 1. Generic PIM.

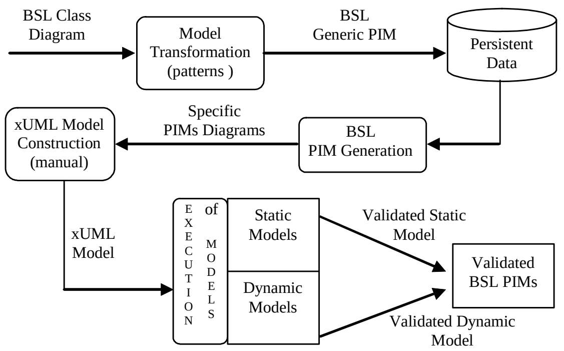

Figure 2. BSL PIM Generation Strategy.

Figure 2 shows the strategy used to create new PIMs for the BSL software.

From a BSL class diagram, a model transformation is applied to obtain a BSL Generic PIM. Model transformation is the process of converting a source model into entrance for the production of a target model as exit [16]. This transformation concept is part of the MDA approach and one of the techniques to carry out the transformation is the use of design patterns. For this transformation, aiming at achieving model generalization, a set of Adaptive Object Model (AOM) [20] design patterns is used.

From the BSL Generic PIM it is possible to organize, in a database, information on class, types of class, attributes, types of attribute, relationships and types of relationship of the specific PIMs to be generated. To manage such information the PIM Generation System (PGS) is used. The PGS, aided by the database, allows the user to create the type of specific PIM which must be generated for a given BSL variation, changing or adding class, attributes and relationships types. The PGS provides an output file which can be converted into a graphic diagram with the use of graphic tools such as ArgoUML [1] by University of California or ESS-Model by Eldean AB [6]. From there, the xUML model is constructed manually using a tool supporting xUML so that the model is executed and verified.

A. Model Transformation

This section explains the transformation carried out, from the BSL class diagram shown in Figure 3, to reach the generic PIM. As mentioned previously, to obtain a generic PIM it is necessary to use the model transformation with the aid of AOM design patterns. The purpose of using such design patterns is to obtain a model as generic as possible, as the set of patterns allows the easy inclusion of new types of objects, changing existing ones and changing the relationships between types. To do so, the following AOM patterns are used: TypeObject, Properties and Accountability.

Figure 4 represents the application of patterns TypeObject and Properties.

TypeObject pattern separates an Entity from an Entity Type, makes an unknown number of the subclasses simple instances of a generic class. With TypeObject it is possible to create new instances of class types. To

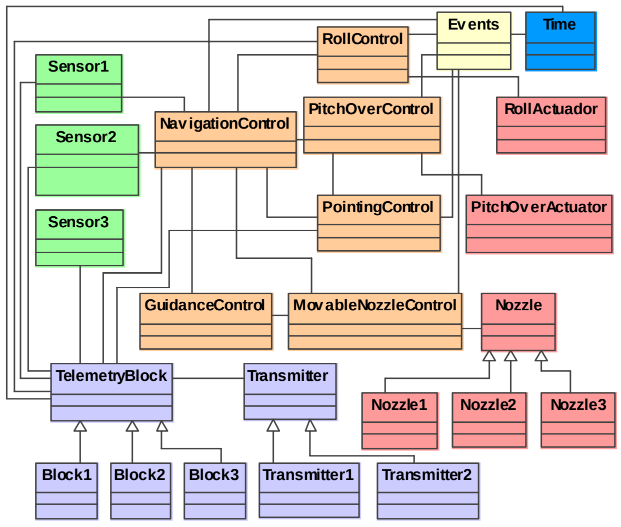

Figure 3. BSL Class Diagram.

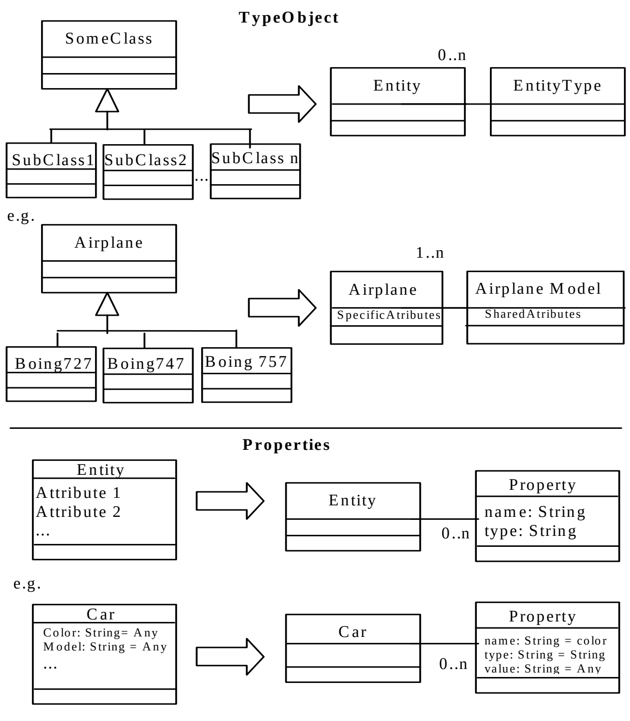

Figure 4. AOM Patterns.

vary the attributes of the new classes generated, the pattern Property is used, as through it each time a new attribute is required, a new class is created. The Properties pattern can define attributes contained in Entities, instead of each attribute being a different instance variable, the Property pattern makes an instance variable that holds a collection of attributes. The pattern Accountability, which manipulates types of relationships, is also used in the transformation model. The creation of the class methods through Strategy pattern is being studied and will be presented in future works. A more complete explanation on the application of AOM patterns may be found in [18] and [20].

As example only classes related to the BSL Telemetry module, part of the class diagram shown in Figure 3, were used. This module can be seen in Figure 5.

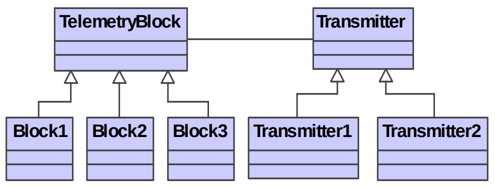

Figure 5. Telemetry Module.

The TypeObject pattern is used to generalize the type of Telemetry Blocks and the types of Transmitters. The Properties pattern is used together with the TypeObject to define attributes and types of attributes of each type of Telemetry Block and Transmitter. In the same way, the Accountability pattern is used together with TypeObject to generalize the types of relationships between classes. In that way, it is possible to obtain a generic model of the BSL Telemetry module.

Figure 6 shows the result of the transformation applying AOM patterns to the classes related to BSL Telemetry module.

Thus, the model becomes generic enough to serve as base for the generation of other specific PIMs. The application of design patterns enables the organization of information concerning the class, types of class, attributes, types of attribute, relationships and types of relationship in the database. It is now possible to manipulate different types of blocks or transmitters with their respective attributes and relationships.

The PGS developed manipulates such data in order to generate new PIMs.

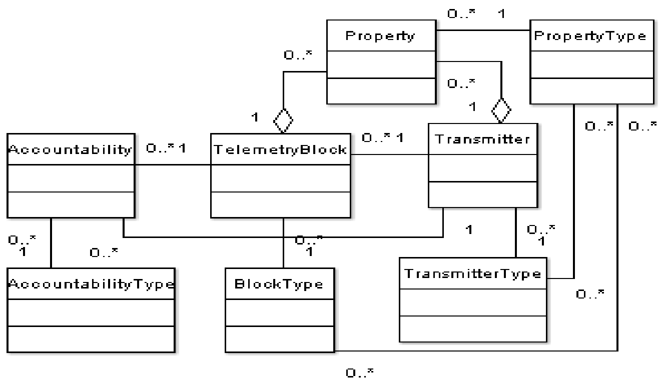

Figure 6. Generic PIM - BSL Telemetry.

B. BSL PIM Generation

The BSL PIM Generation is carried out through the PGS. The PGS is an application developed in Java language, having the function of supplying a java file as a means of enabling the creation of UML graphic diagrams of a specific PIM, through proprietary tools such as ESS Model or ArgoUML.

Figure 7 shows the PGS interface, where the user may define for a given PIM, information on class, attributes and relationships types.

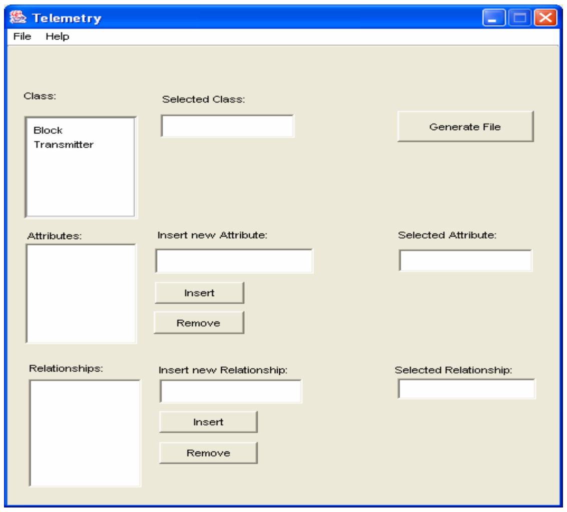

Figure 7. BSL PIM Generator System.

The PGS enables the inclusion or change of types of class, attributes or relationships, according to the user's selection and pre-established rules in the database. The established rules are based on the generated generic PIM. Figure 8 shows a specific telemetry PIM of a BSL variation generated by the PGS, aided by tool ArgoUML. Figure 8 shows that for Telemetry Block class, the objects: Read

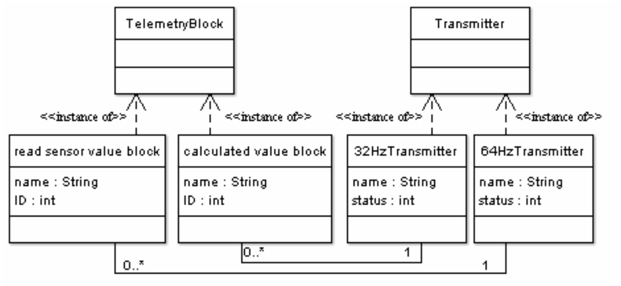

Figure 8. Telemetry PIM Generated.

Sensor Value Block and Calculated Value Block were created and for Transmitter class, the objects: 32Hz Transmitter and 64Hz Transmitter were created. New attributes and new relationships were also included. The next step is to create this model in a tool supporting xUML.

C. Execution of Models

The model generated by PGS must be manually constructed in the xUML modeling tool. The tool selected in this work is iUMLite from Kennedy Carter, a free version which generates xUML models and allows their execution.

Figure 9 shows the Telemetry Use Case created in the iUMLite tool.

Figure 10 shows the transition among states created manually according to the specific PIM generated. The behavior of states is determined by the procedure, written in ASL. For example, in Figure 10, in state Waiting Time, upon receiving event ConstructReadBlock, the FSM fires the transition, arrives at the next state: ConstructingReadBlock, and executes the procedure associated to the new state.

Therefore it is possible to follow up the execution and verify any faults in the models.

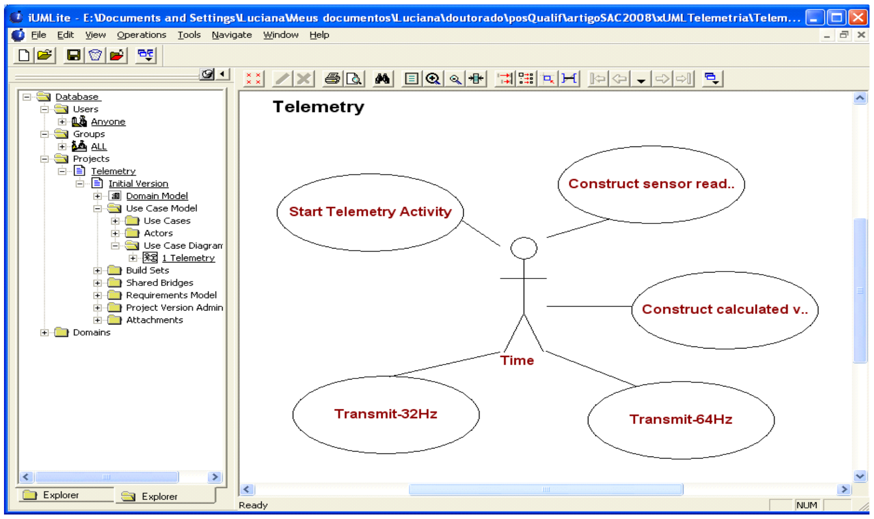

Figure 9. Telemetry Use Case – iUMLite.

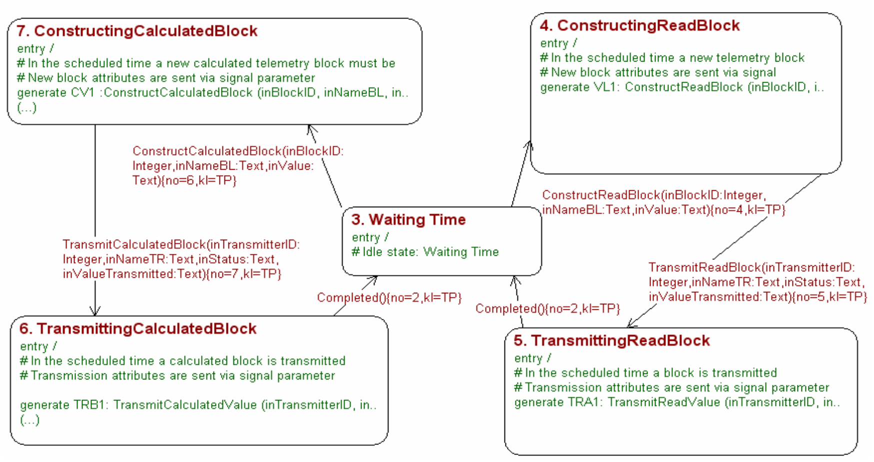
7. ConstructingCalculatedBlock
entry /
# In the scheduled time a new calculated telemetry block must be
# New block attributes are sent via signal parameter
generate CV1 :ConstructCalculatedBlock (inBlockID, inNameBL, in..
(...)
4. ConstructingReadBlock
entry /
# In the scheduled time a new telemetry block
# New block attributes are sent via signal
generate VL1: ConstructReadBlock (inBlockID, i..
ConstructCalculatedBlock(inBlockID:Integer,inNameBL:Text,inValue:Text){no=6,kl=TP}
ConstructReadBlock(inBlockID:Integer,inNameBL:Text,inValue:Text){no=4,kl=TP}
3. Waiting Time
entry /
# Idle state: Waiting Time
TransmitCalculatedBlock(inTransmitterID:Integer,inNameTR:Text,inStatus:Text,inValueTransmitted:Text){no=7,kl=TP}
TransmitReadBlock(inTransmitterID:Integer,inNameTR:Text,inStatus:Text,inValueTransmitted:Text){no=5,kl=TP}
Completed(){no=2,kl=TP}
Completed(){no=2,kl=TP}
6. TransmittingCalculatedBlock
entry /
# In the scheduled time a calculated block is transmitted
# Transmission attributes are sent via signal parameter
generate TRB1: TransmitCalculatedValue (inTransmitterID, in..
(...)
5. TransmittingReadBlock
entry /
# In the scheduled time a block is transmitted
# Transmission attributes are sent via signal parameter
generate TRA1: TransmitReadValue (inTransmitterID, in..

Figure 10. Telemetry FSM.

VII. Final Considerations

The MDA concepts are used in this work as a new development approach for the BSL software, aiming at benefiting from the advantages offered by this architecture, mainly in terms of facilitating changes coming from replacements resulting from technological advancements.

A strategy for the generation of BSL PIM, based on a generic PIM, is presented. The generic PIM discussed is the differentiator in this work, as due to its higher abstraction level serves as base for the understanding of software, and allows the reuse through the obtaining of PIMs specific for other BSL variations. The idea of the generic PIM is also a contribution to state-of-the-art MDA.

The usage of xUML allows the verification to start at the modeling phase. The adoption of xUML enables models to be executed and their behavior to be analyzed so that a more precise modeling is achieved.

References

[1]ArgoUML - http://argouml.tigris.org/

[2]AEB- Agência Espacial Brasileira, Programa Nacional de Atividades Espaciais, Conselho Superior da Agência Espacial Brasileira, Brasília, 2006. http://www.aeb.gov.br/.

[3]Bloomfield, T.; "MDA, Meta-Modeling and Model Transformation". Introducing new Technology into the Defence Industry, *European Conference on MDA* (ECMDA), OMG, 2006.

[4]DeAntoni, J.; Babau, J.; "A MDA-based approach for real time embedded system sismulation", in Proc. *9th Intl Symposium on Distributed Simulation and Real-Time Applications* (DS-RT'05), IEEE, 2005.

[5]Dinh-Trong, T.; Kawane, N.; Ghosh, S.; France, R.; "A tool-Supported Approach to Testing UML Design Models", in Proc.10th *International Conf. on Engineering of Complex Computer Systems* (ICECCS' 05), IEEE, 2005.

[6]EssModel - http://www.essmodel.com/

[7]Fleurey, F.; Steel, J.; Baundry, B.; "Validation in Model_Driven Engineering: Testing Model Transformation", IEEE, 2004.

[8]France, R.; Ghosh, S.; Dinh-Trong, T.; "Model Driven Development Using UML 2.0", IEEE, 2006.

[9]Gao, J.; Li, D.; Zheng, S.; "Developing Real-time System based on Model Driven Architecture", *International Conf. on Mechatronics and Automation*, IEEE, 2006.

[10]Graw, G.; Herrmann, P.; "Verification of xUML specifications in the context of MDA", University of Dortmund, 2002.

[11]Henriksson, A.; Abman, U.; "Improving Software Quality in Safety Critical Applications by Model Driven Verification", Elsevier, 2004.

[12]Houberdon, J.; Babau, J.; "MDA for Embedded System Dedicated to Process Control", *Workshop on Model Driven Architecture in the Specification, Inplementation e Validation of object-Oriented Embedded Systems*, 2003.

[13]Linzhang, W.; Jiesong, Y.; Xiaofeng, Y.; Jun, H.; Xuandong, L.; Guoliang, Z.; "Generating Test Cases from UML Activity Diagram based on Gray-Box Method", 11th *Asia-Pacific Software Engineering Conference* (APSEC'04), 284-291, IEEE, 2004.

[14]Lu, S.; Halang, W.; Zhang, L.; "A Component-based UML Profile to Model Embedded Real-time Systems Designed by the MDA Approach", in Proc. 11th Intl. *Conf. on Embedded and Real-time Computing Systems and Applications* (RTCSA'05), IEEE, 2005.

[15]Morais, P.; "Programa de Veículos Lançadores de Satélites Cruzeiro do Sul - O Brasil na Conquista de sua Independência no Lançamento de Satélites", IAE, CTA, 2005. http://www.aeroespacial.org.br/aab/downloads.php.

[16]OMG, MDA Guide Version 1.0.1, n. omg/2003-06-01, EUA, jun, 2003.

[17]OMG, Model Driven Architecture, n. ormsc/2001-07-01, EUA, jul, 2001.

[18]Riehle, D.; Tilman, M.; Johnson,R.;. "Dynamic Object Model". *Proceedings of Conference on Patterns Languages of Programs* (PloP´2000), 2000.

[19]Santos, W.A.; Loubach, D.S.; Nascimento, M.R.; *Martins, O.A.; Cunha, A.M.; Nobre, J.C.S.; "A PBL Approach for Prototyping MDA Aerospace Software", 15th Educators'* Symposium, ACM, 2006.

[20]Yoder, J. W.; Balaguer, F.; Johnson, R.; "Architecture and Design of Adaptive Object-Models". ACM Sigplan Notices. Vol. 36, Fasc. 12, 50-60, 2001.