

A MODEL-DRIVEN REQUIREMENTS ENGINEERING APPROACH TO CONCEPTUAL SATELLITE DESIGN

Bruno Bustamante Ferreira Leonor, brunobfl@yahoo.com.br

Walter Abrahão dos Santos, walter@dss.inpe.br

National Space Research Institute – INPE
São José dos Campos, SP -Brazil

Stephan Stephany, stephan@lac.inpe.br

National Space Research Institute – INPE
São José dos Campos, SP -Brazil

Abstract. *Satellite systems are growing even more complex, making technical issues a significant driver of costs. The increasing complexity of these systems makes requirements engineering activities both more important and more difficult. Additionally, nowadays strong market competition drive companies to improve the efficiency with which they design and manufacture space products and systems. This imposes a heavy burden on systems-of-systems engineering skills and particularly on requirements engineering which is an important phase in a system's life cycle. A poorly attained requirement engineering approach may cause design failures, cost overrun and delays. One solution is to underpin the preliminary conceptual satellite design with computer-based information reuse and integration to deal with interdisciplinary nature of this problem domain. This can be attained by taking a model-driven engineering approach (MDE), in which models are the main artifact during system development. MDE is an emergent approach that tries to address system complexity by the intense use of models. This work outlines the use of SysML (Systems Modeling Language) for requirements engineering, more specifically requirements management and traceability. It is intended to use this approach in the conceptual phase of a semi-professional satellite system called ITASAT, currently being built by INPE and some Brazilian universities.*

Keywords: *Model-Driven Engineering, ADL, SysML, Satellite.*

1. INTRODUCTION

Space systems are complex systems designed to perform specific functions for a specified design life. Satellite projects, for instance, demand lots of resources, from human to financial, as well accounting for the impact they play on society. This requires good planning in order to minimize errors and not jeopardize the whole mission.

Therefore satellite conceptual design plays a key role in the space project lifecycle as it caters for specification, analysis, design and verification of systems without actually having a single satellite built. Conceptual design maps client needs to product use functions and is where functional architecture (and sometimes the physical architecture) is decided upon.

Moreover, the lack of a clear vision of the satellite architecture hinders team understanding and communication, which in turn often increases the risk of integration issues. The conceptual satellite design phase has been lacking efficient support.

SysML is a new systems modeling language that supports specification, analysis, design, verification and validation of a broad range of complex systems (Soares and Vrancken, 2007). This work proposes to employ SysML as a satellite architecture description language in order to enable information reuse between different satellite projects. Additionally, this approach facilitates knowledge integration and management over systems engineering activities. One of them is requirements engineering, more specifically requirements management and traceability. This is an important phase in the life cycle of satellite systems.

This work shows the main advantages of having user requirements being graphically modeled, their relationships explicitly mapped, and system decomposition considered in the early system development activities. In addition, requirements traceability is enhanced by using the SysML Requirements tables. The approach is illustrated by a list of user requirements for ITASAT, semi-professional satellite system, currently being built by INPE and some Brazilian universities.

This work is organized as follows. Section 2 presents a short introduction to satellites and to SysML as an architecture description language. Section 3 shows the SysML satellite modeling. Section 4 covers the SysML satellite requirements engineering. Section 5 describes further future work. Finally, Section 6 summarizes this research report.

2. BACKGROUND

This background section presents an overview of the satellite and SysML which will be important for the paper context.

2.1. Overview to satellites

A satellite has generally two main parts:

- The bus or platform where the main supporting subsystems reside and;
- The payload, the part that justifies the mission.

As an example, Figure 1 shows the Equatorial Atmosphere Research Satellite (EQUARS), a typical scientific satellite envisaged by INPE (Chamon, 2004). Its various payloads are aimed at studying the equatorial low, middle and upper atmosphere-ionosphere. To achieve this objective, the satellite shall be placed in a near equatorial orbit at 750 km Low Earth Orbit (LEO) altitude, and 20 degrees of inclination, in order to view low latitude regions of the Earth.

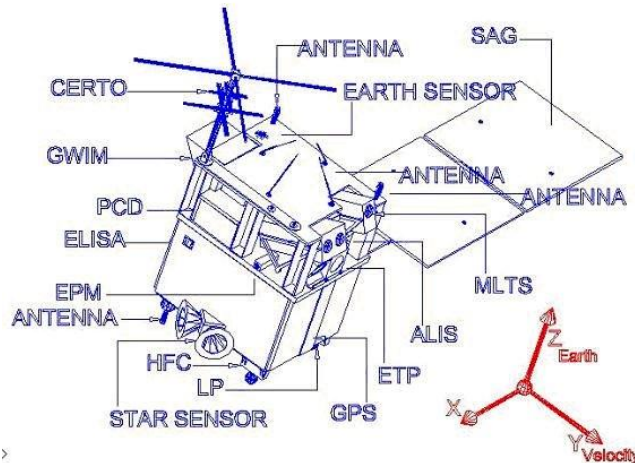


Figure 1. EQUARS satellite (Chamon, 2004)

A typical satellite bus has a series of supporting subsystems as depicted in Figure 2. The satellite system is built around a system bus also called the On-Board Data Handling (OBDH) bus. The main On-Board Computer (OBC) is the bus master which here incorporates functionalities from command and telemetry units for simplicity. The clients on the system bus are the various satellite subsystems and the payloads. Satellite subsystems may have their own computer and may even have an internal bus. The flight software manages all the spacecraft functionalities into a coordinated fashion.

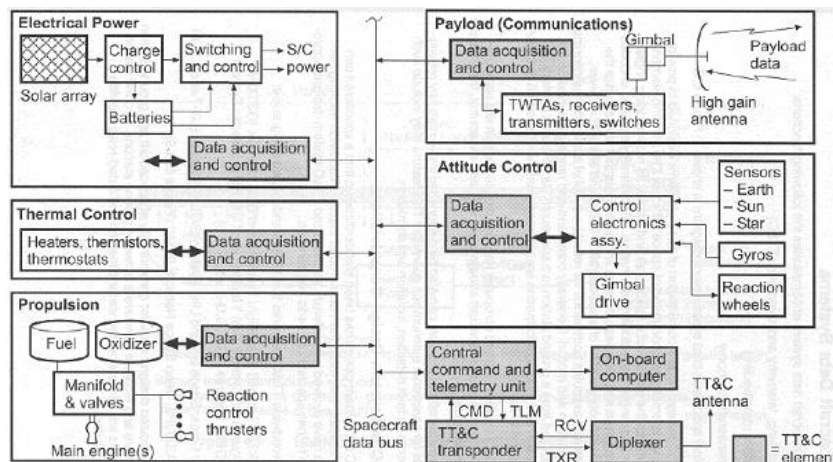


Figure 2. Block diagram of a typical satellite (de Souza, 2002)

Any satellite is composed by subsystems pertaining to either payload or the bus (also called service module or platform). The payload depend on the satellite mission. The control segment on the ground monitors and controls the platform subsystems. These are typically the following: Structure of the satellite:

- Power;
- Propulsion;

- Stabilization and Attitude Control;
- Thermal Control;
- Environmental Control and;
- Telemetry, Tracking and Command.

2.2. SysML as an Architecture Description Language

System modeling based on an architecture description language is a way to keep the engineering information within one information structure. Using an architecture description language is a good approach for the satellite systems engineering domain.

Architectures represent the elements implementing the functional aspect of their underlying products. The physical aspect is sometimes also represented, for instance when the architecture represents how the software is deployed on a set of computing resources, like a satellite.

SysML is a domain-specific modeling language for systems engineering and it supports the specification, analysis, design, verification and validation of various systems and systems-of-systems (SysML, 2009). It was developed by the Object Management Group (OMG) (OMG, 2009) in cooperation with the International Council on Systems Engineering (INCOSE) (INCOSE, 2009) as a response to the request for proposal (RFP) issued by the OMG in March 2003. The language was developed as an extension to the actual standard for software engineering, the Unified Modeling Language (UML) (UML, 2009) also developed within the Object Management Group consortium.

Basically, SysML is used for representing system architectures and linking them with their behavioral components and functionalities. By using concepts like Requirements, Blocks, Flow Ports, Parametric Diagrams and Allocations, it is simple to achieve a profitable way to model systems. A comparison between SysML1.0 to UML2.0 in term of re-use is presented in Figure 3. It summarizes the various diagrams available in SysML (SysML, 2009).

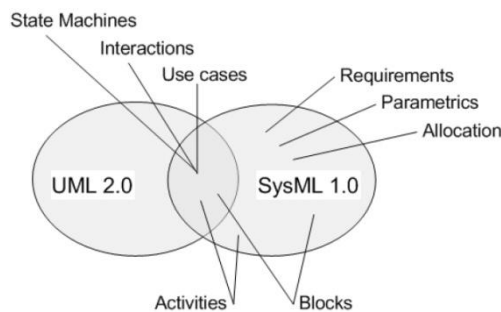


Figure 3. Relationship between UML and SysML (SysML, 2009)

As shown in Figure 4, Requirements, Parametrics and Allocations are new diagrams available only in SysML. Activity and Block diagrams are reused from UML2.0 and extended in SysML. Finally, State Machines, Interactions and Use cases are reused from UML2.0 without modifications.

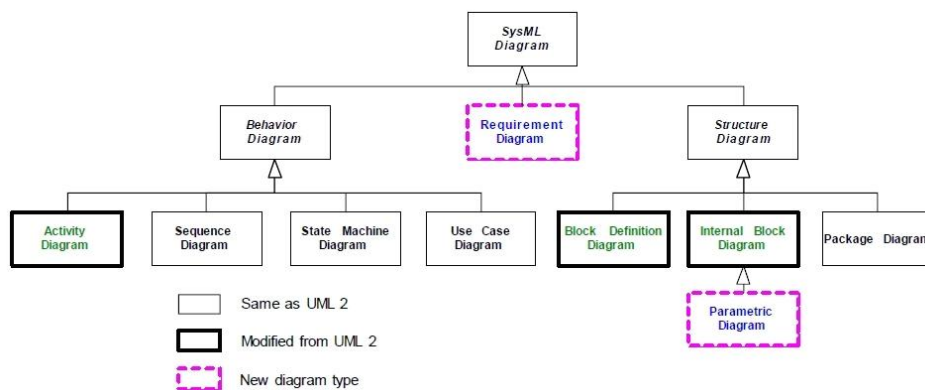


Figure 4. SysML diagram taxonomy (SysML)

This work explores some of the SysML capabilities through an example, the ITASAT student satellite system (Carvalho *et al*, 2008). The application of SysML presented in this work covers only some the diagrams available in SysML due to paper scope and page restrictions.

3. CONCEPTUAL SATELLITE DESIGN VIA SysML

Systems Engineering attacks the problem of design complexity of engineering products as it grows larger, more complex and are required to operate as part of a system.

The approach taken is formal and systematic since the great complexity requires this rigor. Another feature of systems engineering is its holistic view and it involves a top-down synthesis, development, and operation. This suggests the decomposition of the system into subsystems and further into components (Dieter, 1991).

3.1. Motivation for the Satellite SysML Modeling

Space Systems Engineering is a subclass of the previous mentioned in the sense that it is primarily concerned with space systems, e.g. satellite systems. Therefore it deals with the development of systems, including hardware, software, man-in-the-loop, facilities and services for space applications.

The satellite conceptual stage follows the transformation of customer needs into product functions and use cases, and precedes the design of these functions across the space engineering disciplines (for example, mechanical, electrical, software, etc.).

Model-Driven Engineering (MDE) is the systematic use of models as primary engineering artifacts throughout the engineering lifecycle (Schmidt, 2006). MDE can be applied to software, system, and data engineering.

MDE technologies, with a greater focus on architecture and corresponding automation, yield higher levels of abstraction product development. This abstraction promotes simpler models with a greater focus on problem space. Combined with executable semantics this elevates the total level of automation possible. This work argues that MDE is quite suitable for information reuse and integration as will be shown later.

3.2. The SysML Modeling Approach

SysML allows incrementally refinable description of conceptual satellite design and product architecture. This helps systems engineers which are concerned with the overall performance of a system for multiple objectives (e.g. mass, cost, and power). The systems engineering process methodically balances the needs and capabilities of the various subsystems in order to improve the systems performance.

SysML elements in the design represent abstractions of artifacts in the various engineering disciplines involved in the development of the system. The design represents how these artifacts collaborate to provide the product functionalities. The size, volume, and mass constraints often encountered in satellite development programs, combined with increasing demands from customers to get more capability into a given size, make systems engineering methods particularly important for this domain.

This paper explores some of the diagrams available in SysML through the example of the ITASAT satellite system by basically, exploring the block diagram and top-level requirement diagram, both shown in short detail.

SysML diagrams allow information reuse since they can be employed in other similar satellite projects by adaptation and dealing with project variabilities. An exploration of these features for the on-board software design of satellites is shown in (dos Santos, 2008). As an example, take the use case diagram which describes a specific use of the system by a particular actor, i.e., some outside stimulus. The use case diagram represents a fully factored model since use cases are decomposed to find pieces that can be reused in multiple use cases. Its fragments are then composed into use cases using relationships like “extends” and “includes” precluding redundancies.

SysML allows the utilization of use case diagrams which were inherited from the UML without changes (Balmelli, 2007). Use case diagrams are employed to describe systems functionalities by its actors looking forward to achieving a system goal. The use case diagram can be seen as functionalities and/or capabilities which are implemented by means of interaction between actors and the use cases. The use case diagram has been widely applied to specify system requirements.

The interaction between ITASAT actors and some key use cases is shown in Figure 5. This diagram depicts five actors and how they relate to the use cases that they trigger in the high-level system view. The figure still describes schematically the composition of a series of low-level use cases hierarchically modeled by employing an <<include>> dependancy relationship between them. SysML also allows the representation of test use cases which will further explored in the validation, verification and testing project phases. Figure 5 depicts, as an example, the *Test On-Board Management Functions* use case and how are its <<include>> dependancies are related to other two test use cases, *Test Other On-Board Functions* and, *Test Power Supply Functions*.

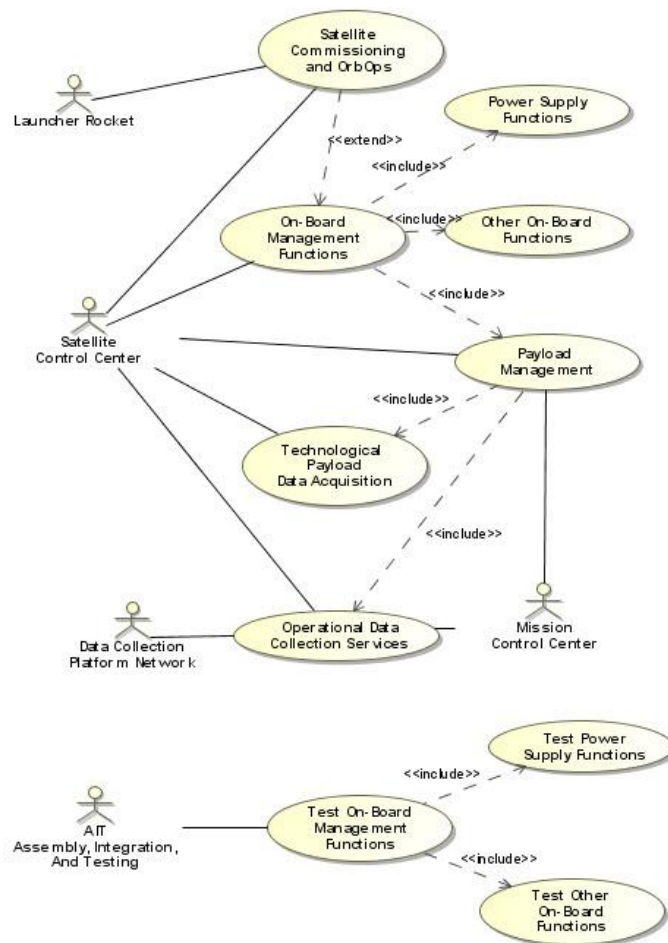


Figure 5. ITASAT high-level use case to specify system requirements (Carvalho *et al*, 2008)

The SysML block diagram is used to show features and high-level relationships. It is used to allow systems engineer to separate basically the responsibilities of the hardware team from the software team. Figure 6 shows the various ITASAT blocks and their interdependencies.

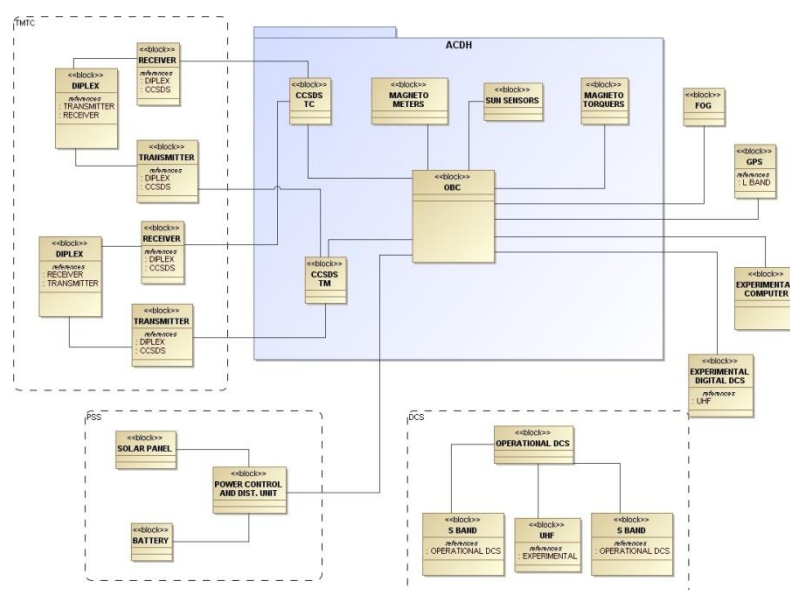


Figure 6. The ITASAT satellite SysML block diagram (Carvalho *et al*, 2008)

The requirements diagram plays a key role into the SysML model as requirements present in this diagram can also appear in other SysML diagrams linking the problem and solution spaces. Furthermore, the requirements diagram notation provides a means to show the relationships among requirements including constraints, see Figure 7 for details. This topic is of high importance to this work hence it is further developed in the next section.

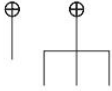
Diagram Element	Notation	Description
Containment Path		The containment relationship is used to represent how requirements are contained in specifications (packages), or how a complex requirement can be partitioned into a set of simpler requirements without adding or changing their meaning.
Derivation Path	«deriveReq»	A derive relationship occurs between a source requirement and a derived requirement, based on analysis of the source requirement.
Satisfaction Path	«satisfy»	A satisfy relationship is used to assert that a model element corresponding to the design or implementation satisfies a particular requirement.
Verification Path	«verify»	A verify relationship is used between a requirement and a test case or other named element to indicate how to verify that the requirement is satisfied.
Refinement Path	«refine»	The refine relationship is used to reduce ambiguity in a requirement by relating it to another model element that clarifies the requirement.
Trace Path	«trace»	A trace relationship is a general-purpose way to relate a requirement and any other model element, useful for relating requirements to documents, etc.
Copy Path	«copy»	The copy relationship relates a copy of a requirement to the original requirement, to support reuse of requirements.

Figure 7. Relationships used in a SysML requirement diagram (OMG–SysML, 2009)

4. THE MODEL-DRIVEN REQUIREMENTS ENGINEERING APPROACH

The process of requirements engineering involves various key activities, such as elicitation, specification, prioritization and management of requirements. This section shows how SysML can enable a model-driven requirements engineering approach to conceptual satellite design.

The SysML standard identifies relationships that enable the modeler to relate requirements to other requirements as well as to other model elements. These include relationships for defining a requirements hierarchy, deriving requirements, satisfying requirements, verifying requirements, and refining requirements (SysML, 2009).

Figure 8 shows a simplified view of the ITASAT requirement tree structure (Carvalho *et al*, 2008). It also shows how a constraint is attached to a low-level requirement and how traceability may be established.

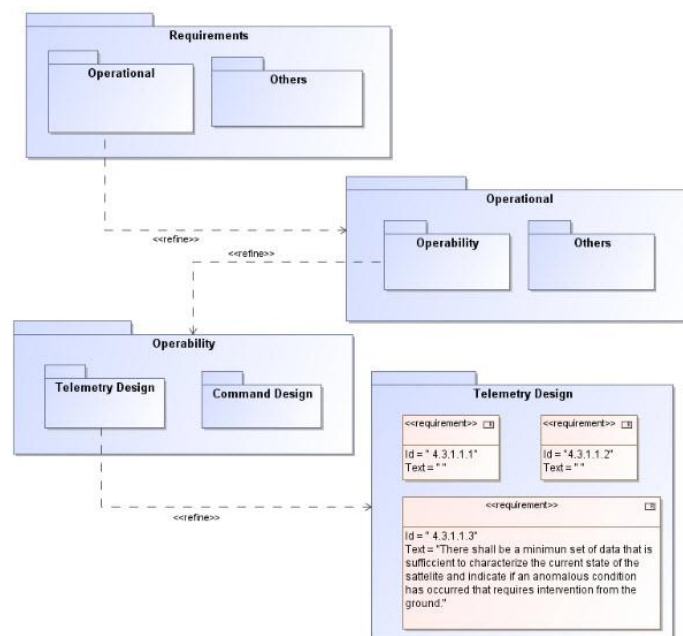


Figure 8. Requirements tree structure for the ITASAT satellite (Carvalho *et al*, 2008)

After top-level requirements are elicited, then starts the decomposition of every system requirement into progressively lower levels of design. This is done by defining the lower-level functions which determine how each function must be performed. Allocation assigns the functions and its associated performance requirements to a lower level design element. Decomposing and allocation starts at the system level, where requirements derive directly from mission needs, and proceeds through segment, subsystem, and component design levels (Larson and Wertz, 2004). This process must also warrant closure at the next higher level meaning that satisfying lower-level requirements warrants performance at the next level. Additionally, it roundtrips all requirements tracing back to satisfying mission needs.

Managing requirements is the capability of tracing all system components to output artifacts that have resulted from their requirement specifications (forward tracing) as well as the capability of identifying which requirement has generated a specific artifact or product (backward tracing) (Pressman, 1995).

The great difficulty on tracing requirements is responding the following questions: *What to track?* and *How to track?*. One can say that a requirement is traceable when it is possible to identify *who has originated it, why it exists, which are the requirements related to it? how is it related to other project information*. These information is used to identify all requirement\elements affected by Project changes. The specification of requirements can facilitate the communication between the various project stakeholder groups.

There are several published works on requirement engineering and the most common way they employ to requirement tracking is by posing basic questions about the underlying domain (Aurum, 2005). Unfortunately, such questionnaire does not offer generally any classification on the sufficient elements in order to identify all model elements.

By using a SysML requirements diagram, system requirements can be grouped, which contributes to enhance project organization showing explicitly the various relationship types between them (Soares and Vrancken, 2008). These include relationships for defining requirements hierarchy or containment, deriving requirements, satisfying requirements, verifying requirements and refining requirements (OMG-SysML, 2009). Moreover, the SysML requirements diagram can be employed to standardize how requirements are documented following all their possible relationships. This can provide systems specification as well as be used for requirements modeling.

New requirements can be created during the requirement analysis phase and can be related to the existing requirements or complement the model. Figure 9 presents an excerpt from the ITASAT requirements diagram which utilizes the `<<deriveReq>>` relationship type showing the derived *Satellite State* requirement from the source *Telemetry Design* requirement inside the Operability requirement SysML package. This allows, for example, a link between high-level (user oriented) and low-level (system oriented) requirements which contributes to explicitly relates the dependency of user requirements mapped into systems requirements.

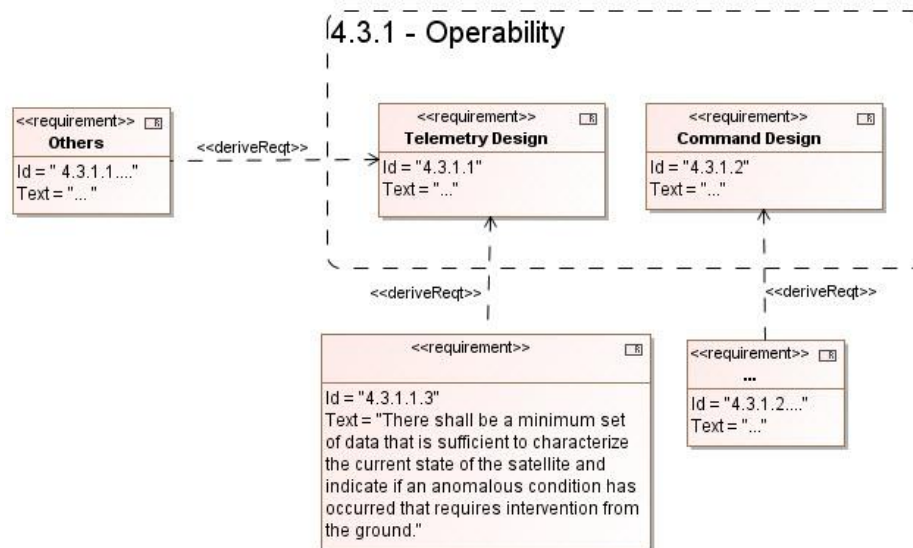


Figure 9. An excerpt of the ITASAT requirements diagram with a `<<deriveReq>>` relationship (Carvalho *et al*, 2008)

Similarly, Figure 10 presents another excerpt from the ITASAT power subsystem requirements diagram which utilizes three relationships. Requirements are abstract classes with no operations neither attributes. Subrequirements are related to their “father” requirement by utilizing the `<<containment>>` relationship type. This is shown in Figure 10 as many subrequirements from the *Power Supply Requirements* requirement are connected employing `<<containment>>` relationships. The “father” requirement can be considered a package of embedded requirements.

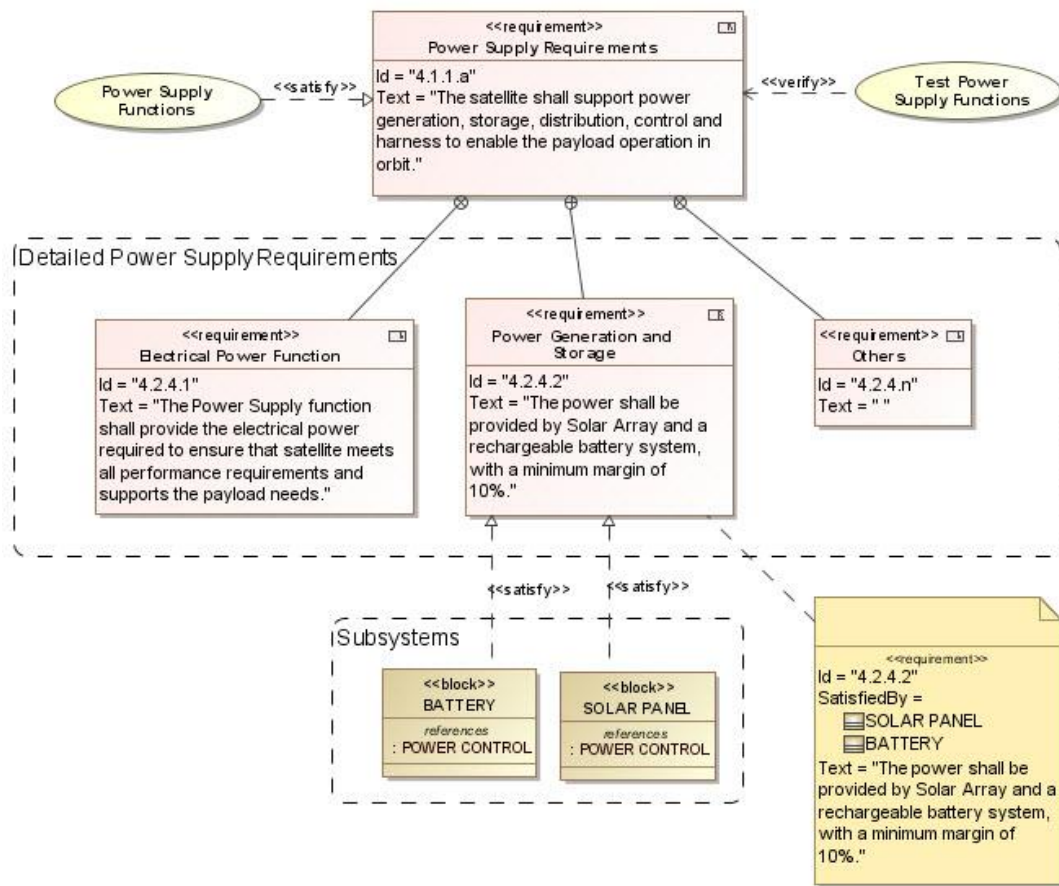


Figure 10. An excerpt of the ITASAT power subsystem requirements diagram with containment `<<satisfy>>` and `<<verify>>` relationships (Carvalho *et al*, 2008)

Additionally, Figure 10 presents the `<<satisfy>>` relationship type which shows how a model satisfies one or more requirements. It represents a dependency relationship between a requirement and a model element, in this case the *Power Supply Functions* use case is satisfied by the *Power Supply Requirements*. Finally, it is shown the `<<verify>>` relationship type where the *Test Power Supply Functions* test use case is verified by the functionalities provided by the *Power Supply Requirements*. This may include standard verification methods for inspection, analysis, demonstration or test.

Lastly, SysML allows requirements traceability by using tabular notations. This allows model elements to be traced in SysML via requirements tables which may contain fields like: identifier (ID), name, which requirement is related to it, what type of relationship is held among them.

One such SysML tabular notation for requirements traceability is shown in Figure 11 which is suitable for crossrelating model elements. The figure shows a requirement matrix table where cross-tracing is done between requirements, blocks defined in the ITASAT block diagram and high-level use cases. This table is quite important as it allows requirements traceability towards issues like:

- Understanding of requirements origin;
- Project scope management;
- Requirements change management;
- Impact analysis of requirement changes;
- Impact analysis of requirement test failures (i.e., if a test fails then a requirement may not be met);
- Verification if all system requirements are mapped into implementation and;
- Validation if the final application only performs what it was previously expected.

	CCSDS TC [AC...	CCSDS TM [A...	MAGNETO ME...	MAGNETO TO...	OBC [ACOH]	SUN SENSORS...	BATTERY	Determine the ...	DIPLEX	EXPERIMENTA...	EXPERIMENTA...	FOG	Generate Report	GPS	Harness	L BAND	On-Board Man...	Open XML File	Operational Da...	OPERATIONAL ...	OperationalDCS	Other On-Boar...	Others	Parse Paramet...	Payload Manag...	Power	POWER CONT...	Power Supply ...	Processo Busine...	RadiationEffectExp	RECEIVER	S BAND	S BAND TM/TC	SOLAR PANEL	Structure	Technological ...	Test On-Board...	Test Other On...	Test Power Su...	ThermalControl	TMTC	
Data							3																				3															
Harness																																										
Power Supply Require...																											X													X		
Reability																																										
S Band																																										
Storage																																										
Structure																																										
Telecommand Receiver																																										
Telemetry Transmitters																																										
Thermal Control																																										
Transportation																																										
UML Standard Profile																																										
Requirements																																										
Operational [Requir...																																										
Operability [Requ...																																										
Others [Require...																																										
Power Supply Require...							3																																			
Electrical Power Fun...																																										
Others [Power Supp...																																										
Power Generation a...								X																																		
Electrical Power Fun...																																										
Battery [Power S...																																										
Operability																																										

Figure 11. The tabular matrix notation used to display power related requirements and their relationships to other model elements (Carvalho *et al*, 2008)

Additionally, requirements can be traced also by navigating through SysML requirement diagrams on the anchors points shown in Figure 10 by means of stand-out notes. The anchors contain information like the relationship type and with which model element the requirement is related and vice-versa, given a model element it may reference all requirements related to this element. Doing so, it allows a quick and simple way to identify, prioritize and improve requirements traceability.

The resources provided by the SysML are by far beyond the capabilities here presented due to paper page limitation.

5. FUTURE WORK

Currently more complete ITASAT SysML modeling is also expected which include:

- Enhancing Block Diagram representation to model detailed subsystems and components, and ports describing their interfaces;
- Checking dependencies (e.g. analytical) between structural properties expressed using constraints and represented using the parametric diagram;
- Exploring features behavior modeling, namely interactions, state machine and activities;
- Employing SysML for providing a mechanism to relate different aspects of the model and to enforce traceability across it;
- Development of a novel knowledge-based software tool, named SatBudgets, is being performed to support preliminary budgetings on conceptual satellite design which demands interdisciplinary skills and;
- Provide an interface for Eclipse IDE aggregation as well as for docking to an in-house Satellite Simulator.

6. CONCLUSIONS

Space systems requires strong systems engineering to deal with systems-of-systems complex issues, manufacturing demands and keep risks manageable. Requirements Engineering for these systems is a difficult issue as when this is poorly performed, various problems may occur, such as failures, cost overrun and delays. A case study was presented in this work introducing the use of SysML satellite modeling for requirements engineering. This allows user requirements being graphically modeled, their relationships being explicitly mapped and, system decomposition being considered in the early system development activities. By employing SysML as a satellite architecture description language, it also enables information reuse between different satellite projects as well as it facilitates knowledge integration and management on systems engineering activities. This work will be further extended to implement MDE automation concepts into the ordinary workflow of satellite systems engineering.

7. REFERENCES

Aurum, A. W., 2005, "Engineering and Managing Software Requirements". Springer Verlag, New York.

- Balmelli, L., 2007, "An overview of the systems modeling language for products and systems development", *Journal of Object Technology*, vol. 6, No. 6, pp. 149-177.
- Carvalho, T. and et al., 2008, "Itasat satellite specification. Technical report", INPE U1100-SPC-01 Internal Report.
- Chamon, M., 2004, "Equars satellite specification. Technical report", INPE E30000-TSD-042 Internal Report.
- de Souza, P. N., 2002, "Space Project Management", CITS lecture notes. Slides - INPE – Brazilian National Space Research Institute.
- Dieter, G. E., 1991, "Engineering Design - a Materials and Processing Approach", McGraw-Hill International Edition, New York.
- dos Santos, W. A., 2008, "Adaptability, Reusability and Variability on Software Systems for Space On-Board Computing", ITA Ph.D. Thesis, S. J. Campos, Brazil.
- INCOSE - International Council on Systems Engineering, 2009, <http://www.incose.org>.
- Larson, W. J., Wertz, J. R., 2004, "Space Mission Analysis and Design". McGraw-Hill, El Segundo, USA, 3rd edition.
- OMG - Object Management Group, 2009, <http://www.omg.org>.
- OMG-SysML, 2009, "SysML 1.0 Specification", <http://www.omg-sysml.org/>.
- Pressman, R. S., 1995, "Engenharia de Software", Makron Books.
- Schmidt, D. C., 2006, "Model-driven engineering", *IEEE Computer*, vol. 39, No. 2, pp.25–31.
- Soares, M. dos S., Vrancken, J., 2007, "Requirements specification and modeling through sysml", *Proceedings of IEEE-SMC*, pp. 1735-1740.
- Soares, M. dos S., Vrancken, J., 2008, "Model-driven user requirements specification using sysml", *Journal of Software*, vol. 3, No. 6, pp. 57-68.
- SysML, 2009, "System Modeling Language", <http://www.sysml.org>.
- UML, 2009, "Unified Modeling Language", <http://www.uml.org>.

8. RESPONSIBILITY NOTICE

The authors are the only responsible for the material included in this paper.