Specifying Satellite Behavior for an Operational Simulator

Jun Tominaga[(1)], Christopher Cerqueira[(1)], Janio Kono[(1)], Ana Ambrosio[(1)]

Brazilian Institute for Space Research - INPE[(1)]
Av. dos Astronautas, 1758
12.227-010 – São José dos Campos – SP - Brazil
E-mail: jun@ccs.inpe.br , christophercerqueira@gmail.com , kono@dss.inpe.br , ana.ambrosio@inpe.br

ABSTRACT

To write a satellite simulator specification for a relatively simple satellite may be simple as the information can be shared among the small group of hardware experts, system engineers and software programmers. However, this becomes extremely difficult as the number of people involved increases. This paper focuses on questions related to how to write effective simulator specification documents to describe the behavior of complex satellites, in such a manner that the specification be comprehensive and clear to all the people involved in the simulator project development. Subsystem designers, system architects, electrical, mechanic and software engineers, as well as mathematicians and programmers with a diverse assortment of backgrounds and different points of view are supposed to work together towards a common goal, which is the development of an operational satellite simulator. Consequently, the greatest challenge to build a satellite simulator is to find a common language, understandable and acceptable by every stakeholder, yet powerful enough to describe the satellite behavior in sufficient detail. The current existing standards for satellite simulators were found to lack some important details concerning how to write specifications of the models, in a precise manner. This paper presents the representation of the knowledge about the subsystems behavior in every its states through well-organized tables. This solution aimed to reduce the gap in the language to represent the satellite behavior in the Specification of the Operational Simulator of the CBERS 3&4 Satellite being developed at INPE.

INTRODUCTION

One of the greatest challenges in developing an operational satellite simulator consists of specifying the behavior of the satellite to be represented in the final software product. A compromised solution must be achieved from a tradeoff between a high fidelity model with extreme complexity and a low-cost simpler implementation with a poor performance.

Currently, INPE is engaged in the development of an operational simulator of the CBERS-3&4 satellites, named SIMC3 [1] [2]. The specifications of the SIMC3 include requirements for a high-fidelity model for the electrical power consumption and the communication interfaces with the Brazilian Satellite Control System. Taking into account the difficulties in providing the huge amount of specifications and the hard work to manage the communication interfaces required for the development of this operational satellite simulator, associated with a short timeframe, the adopted development philosophy was based on prototyping and incremental coding. Once the software architecture and the kernel of the simulator are well defined, the

difficulties now are on writing the subsystem model specification. Recently, two different teams started working in parallel, one taking care of the software solutions while the other is in charge of providing the satellite behavior specification.

The way the satellite subsystems behavior is currently documented represents the best solution found up to now, in representing the information obtained from engineers responsible for each satellite subsystem in a clearly understandable manner the software development team would understand. Several different formats were proposed and refined until a consensus solution agreeable among all persons involved was found at last. Due to project constraints, the knowledge of available hardware design engineers is restricted to the subsystem level. Therefore, the system behavior information has to be modularized into subsystem models and will be initially validated independently. At the end of this entire process, software programmers are expected to receive reliable and easily implementable information of the satellite behavior. Moreover, in order to decrease the learning curve, Subsystem Model Specifications have been documented as uniformly as possible.

An overview of the Simulator development process and the Subsystem Model Specification organization are presented in next sections.

SIMULATOR DEVELOPMENT PROCESS

For each satellite subsystem, writing the Subsystem Model Specification document is a highly demanding enterprise that requires coordinated efforts made by several stakeholders. The involved stakeholders are: Project Manager, Simulator Architect, Subsystem Engineers, Modelers and Software Engineer. The characterization of each stakeholder adopted in this process is based on the definition given in reference [3]. A static view of their roles and the products they deliver are shown in Figure 1.

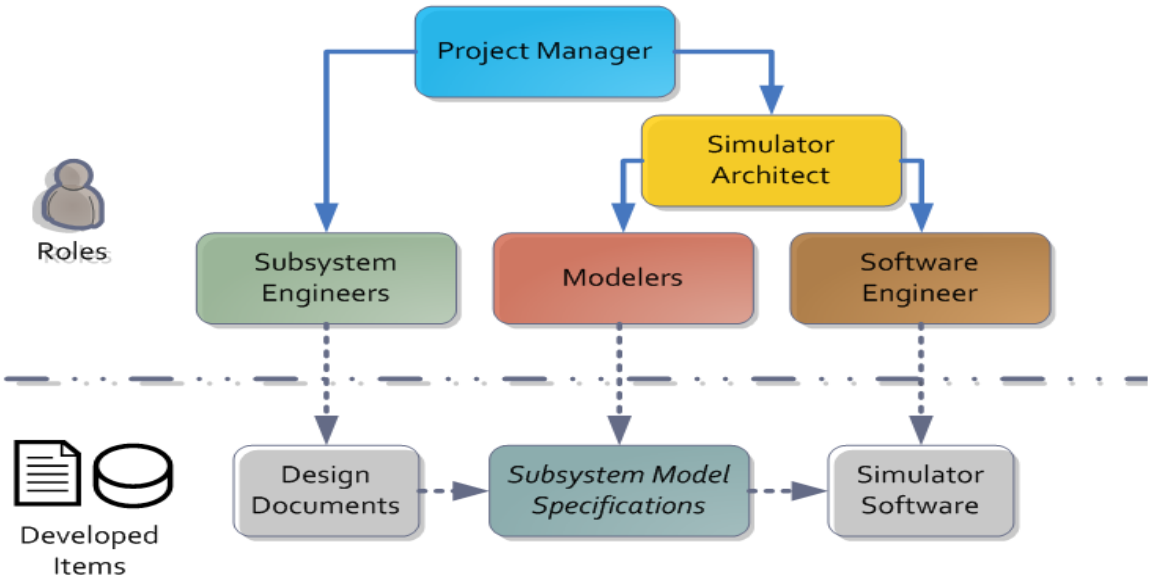

Fig. 1. SIMC3 Development Teams and Developed Artifacts.

The *Software Engineer* is responsible for designing and building the Simulator Software and managing the team of software developers for this purpose. She works with Simulator Architect to make sure that the satellite behavior described by the Subsystem Model Specifications is implemented properly. In order to be able to implement the Simulator Software, the Software Engineer needs the Satellite Specifications, including the Subsystem Model Specifications.

Modelers are the people who write the Subsystem Model Specification documents, based on information gathered from Subsystems Design Documents. They are tasked with mapping and reformatting the subsystems behavior information, under close supervision of the Simulator Architect. Every time the Modelers finish a Subsystem Model Specification, their work is submitted to a Review Meeting for assessment.

The *Simulator Architect* is responsible for the overall architecture of the Simulator Software. She also facilitates the information exchange between the other teams. She is responsible for the configuration control of Subsystem Model Specifications. Upon receiving a new Subsystem Model Specification from the Modelers, she submits it to the Program Manager for analysis at the next Review Meeting. Once approved, the Simulator Architect sends the new Subsystem Model Specification to the Software Engineer for implementation. If Modelers find it necessary, the Simulator Architect may also arrange interviews with Subsystem Engineers for discussions on subsystems features to be implemented. She shall to assure the simulator will run according to the specification, so she will be responsible for the acceptance tasks.

Subsystem Engineers are system engineers regarding the design and implementation of the satellite subsystems for which they are responsible. They hold the most accurate and updated information regarding the working behavior of their subsystems and, as such, are the ones who are able to point out inaccuracies in Subsystem Model Specifications.

The *Program Manager* is the overseer of the satellite mission program. He has the overall view of the satellite system and manages the activities of all other teams. He opens and chairs all Review Meetings for the purpose of evaluating Subsystem Model Specifications. Review Meetings are where the Program Manager, the Simulator Architect, Modelers and Subsystem Engineers evaluate a newly written Subsystem Model Specification. Depending on the agreement reached in the Review Meetings the Simulator Architect arrange other meetings up to the models are acceptable and correct.

Design Documents include hardware specifications, interface datasheets, and handbooks written by manufacturers responsible for the delivery of each contracted subsystem, under the supervision of the Subsystem Engineers. As such, these documents describe the satellite working behavior at subsystem level. Those documents hold all the basic information that the Modelers team needs to consult in order to generate a comprehensive list of parameters that must be represented in the subsystem model.

Subsystem Model Specifications contain all relevant information harnessed by Modelers from the Design Documents, plus eventual interviews held with Subsystem Engineers that were considered essential for describing the behavior of the satellite subsystems, reformatted appropriately to facilitate the coding of the Simulator Software.

The Simulator Software is the final product of the satellite simulator development. According to the general architecture of the Simulator Architect, the Simulator Software must consist of a modularized piece of

software including a core engine (kernel) attached with several reconfigurable modules, to be updated as necessary. Actual design and implementation of these modules is under the responsibility of the Software Engineer, provided they follow the behavioral guidelines set by the Subsystem Model Specifications.

A dynamic view of the development teams and the artifacts they produce are shown in Figure 2, which summarizes the overall workflow of the satellite simulator development process. The delivery of the Design Documents by Subsystem Engineers is the first step. Following this, the Subsystem Model Specification is written by Modelers, based on the analysis of Design Documents and interviews with Subsystem Engineers. Once finished, the Subsystem Model Specification is forwarded to the Review Meeting, from where a Review Report might be issued or the Subsystem Model Specification accepted (transition arc shown in blue). If accepted, the Subsystem Model Specification is forwarded to the Software Engineer for implementation. Otherwise, the Subsystem Model Specification, and possibly the Design Documents also, is updated until the Subsystem Model Specification is accepted.

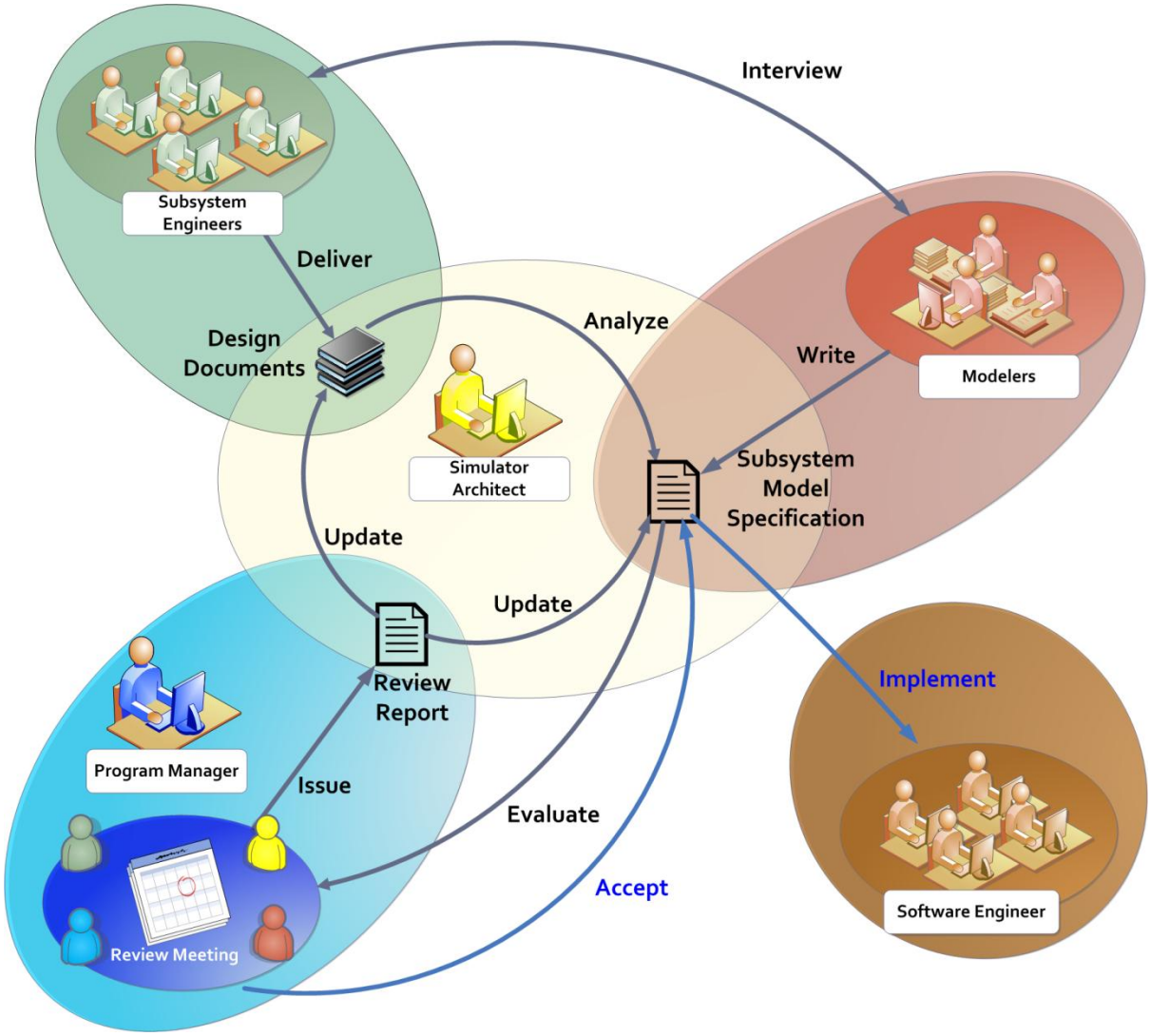

Fig. 2. SIMC3 Development Workflow.

The following section presents how information obtained from the Design Documents is translated into Subsystem Model Specifications representing the behavior of physical subsystems.

SUBSYSTEM MODEL SPECIFICATION OVERVIEW

The main goal of the Subsystem Model Specification is to document the satellite subsystem behavior in such a manner that would make it easy for the Software Engineer team to implement and, at the same time, not difficult for Subsystem Engineers to validate it. Each Subsystem Model Specification document was thus divided into three sections. The presented approach is detailed in [4].

The first section describes the subsystem according to the Subsystem Engineers understanding. Subsystem information is presented here as physical, logical and power views, the latter being further subdivided into electrical, thermal, and radio-frequency communication sub-views.

In the second section, all relevant simulation parameters are listed in tabular form. This works basically as a compilation of all relevant parameters, properly identified so as to avoid confusion or misunderstanding. Input parameters include telecommands and external interferences. Output parameters include switch configurations, working states, power figures, telemetry values and operating modes.

The third section presents the rules describing the subsystems dynamic behaviors in tabular form. A rule is evaluated by testing the precondition, in order to decide whether associated effects are to be applied or not. Columns here are grouped into preconditions and effects. If all precondition parameters of a row satisfy the triggering criterion, then effects are applied as specified on the corresponding row.

Thus, based on their content, the three sections are also referred to as the Subsystem Description section, the Model Parameters section, and the Model Behavior section.

The Subsystem Description section describes the subsystem according to the different views and sub-views. The concept of views was adopted in order to describe functionally similar aspects in different subsystems. Figure 3 illustrates the breakdown of the satellite into subsystems and views.

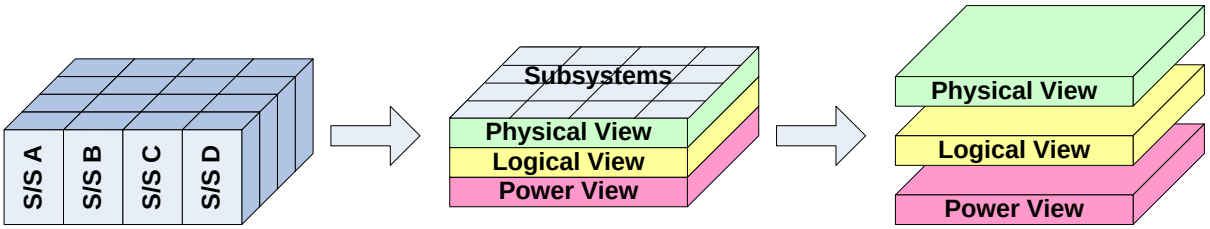

Fig. 3 Satellite Breakdown into Subsystems and Views.

Whereas the division of the system into subsystems allows an engineer to study and implement smaller and simpler units, expressing it in terms of views allows a software expert to understand and implement each subsystem as a data unit with several functionalities in common. Under this approach, the models can be implemented in a progressive manner, by coding first the features considered most important

by the user. Figure 4 shows an example of how models can be progressively improved as new simulator versions are issued.

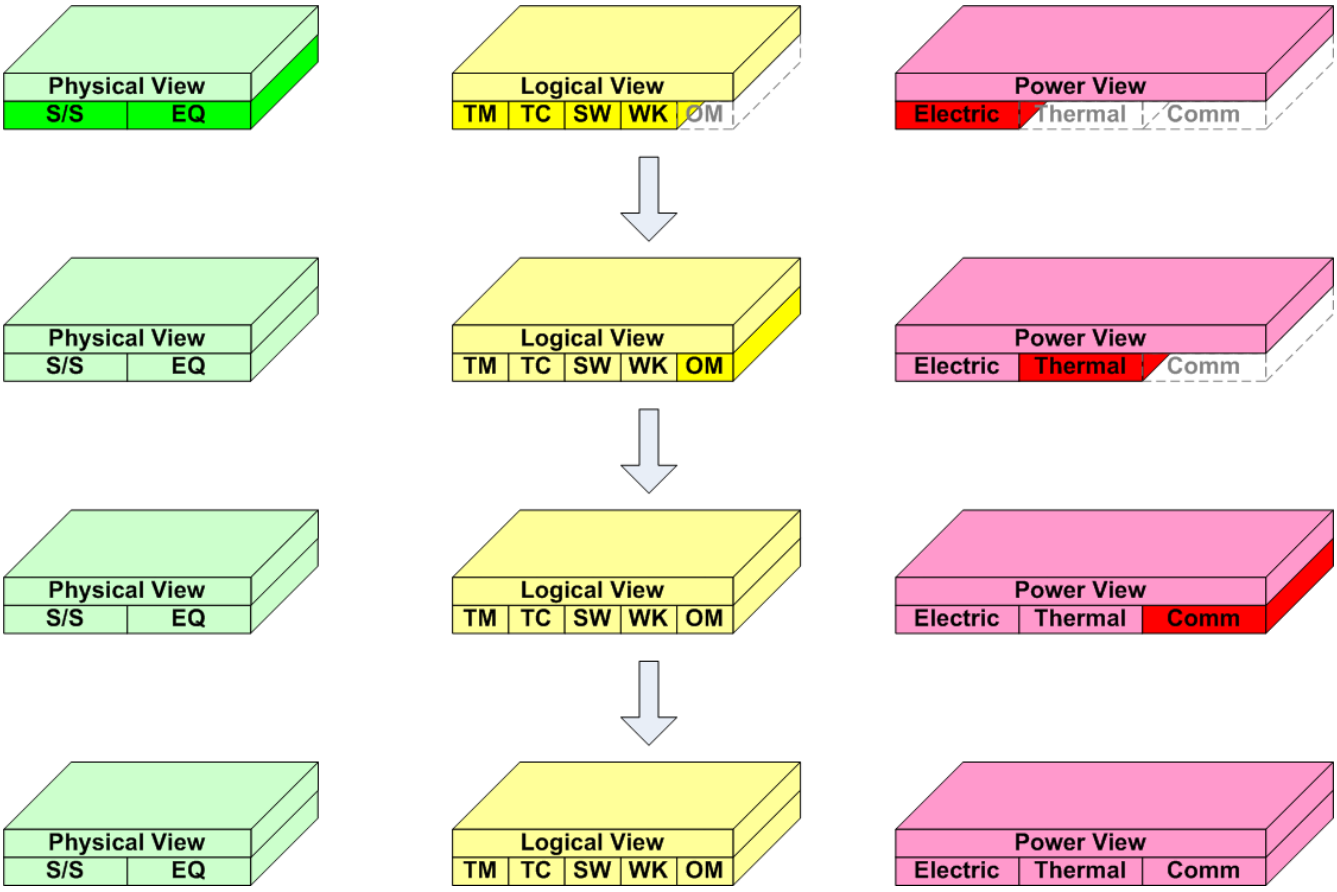

Fig. 4. Example of Simulator Implementation Strategy.

The Physical View describes the subsystem (S/S) in terms of their constituting equipment (EQ), units, parts and components, and their respective mounting locations. This view shows the physical building boxes of each subsystem and assigns a unique name to each of them, following the naming convention specified in the Design Documents. This is usually shown as a tree listing of parts followed by a block diagram.

The Logical View describes the logical states and transitions that define the status of the subsystem at any given time. This view usually shows a logical diagram that presents state-defining logical switches (SW), telecommands (TC) that change their configurations, and telemetries (TM) used to monitor them. It also contains lists of valid operational modes (OM) defined at subsystem (S/S) and equipment (EQ) levels, as well as the definition of working states (WK) parameters necessary to provide a complete mapping of valid, invalid, foreseen and unforeseen operating modes.

The Power View describes the subsystem in terms of power distribution and energy balance. This view is further expanded into Electrical, Thermal, and Communication Sub-Views. Generally speaking, electrical power generated by solar panels is distributed and consumed by subsystems equipment from the main bus and DC/DC converters. Most of the consumed electrical power is dissipated as heat, but a fraction is irradiated as

RF signals. The amount of simulation parameters defined at this view and the complexity of the behavior rules associated with them allow for more or less realistic modeling of analog and thermal telemetry.

Figure 5 summarizes the elements that comprise the Subsystem Description section. It functions as a guideline to be followed by all Subsystem Model Specification documents regarding the contents of this section.

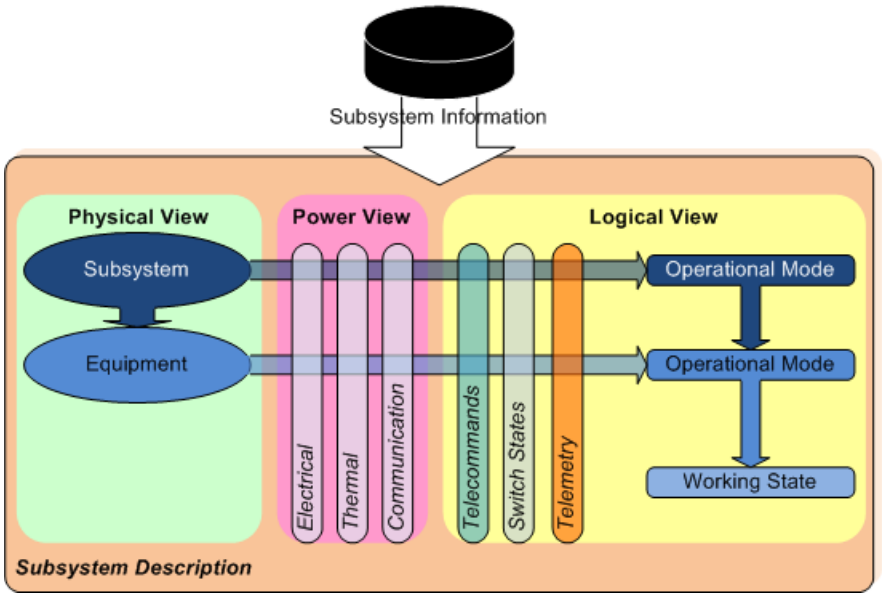

Fig. 5. Subsystem Description Elements.

The Model Parameters section contains a compilation of all relevant simulation parameters. Two major groups of parameters have been identified. Input parameters include telecommands and external events that provoke changes in the behavior of the subsystem model. Output parameters are those affected as consequences of input parameters and value changes in other output parameters. Output parameters include switch configurations, working states, power figures, telemetry values and operating modes.

Figure 6 summarizes the elements that comprise the Model Parameters section. It functions as a guideline to be followed by all Subsystem Model Specification documents taking into account the contents of this section.

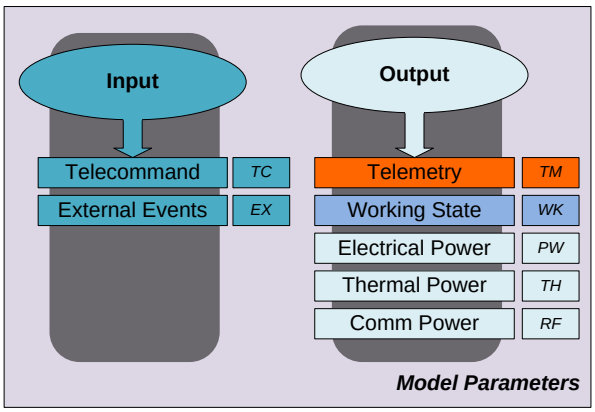

Fig. 6. Model Parameters Elements.

The Model Behavior section contains the rules describing the dynamic behavior of the subsystem. They are essentially causal clauses based on preconditions, formed by logical expressions formed by input and output parameters, which triggers an effect event. Historically these rules used to be written in an algorithmic format that was much closer to the final software implementation, but this has changed since Subsystem Engineers responsible for their analyses alleged that this approach made validation difficult. After several rounds of discussion in meetings, it was decided that these rules were best represented in tabular form. According to the current standard, each rule is represented by a set of clauses defined by a common set of precondition and effect parameters. Figure 7 illustrates an example of rule represented in the current tabular form and its analogue in algorithmic notation.

Preconditions			**Effects**		
Param A	**Param B**	**Param C**	**Param X**	**Param Y**	**Param Z**
a1	b1	c1	x1	y1	z1
a2	b2	> c2	x2	y2	z2
<= a3		-	x3	-	Z+C

Equivalent to:

```
if (A == a1 and B == b1 and C == c1)
   then (X = x1, Y = y1, Z = z1)
if (A == a2 and B == b2 and C > c2)
   then (X = x2, Y = y2, Z = z2)
if (A <= a3 and B == b2)
   then (X = x3, Z = Z+C)
```

Fig. 7. Example of Model Behavior Rule Representation.

Figure 8 presents the elements that comprise the Model Behavior section. It functions as a guideline to be followed by all Subsystem Model Specification documents regarding the contents of this section.

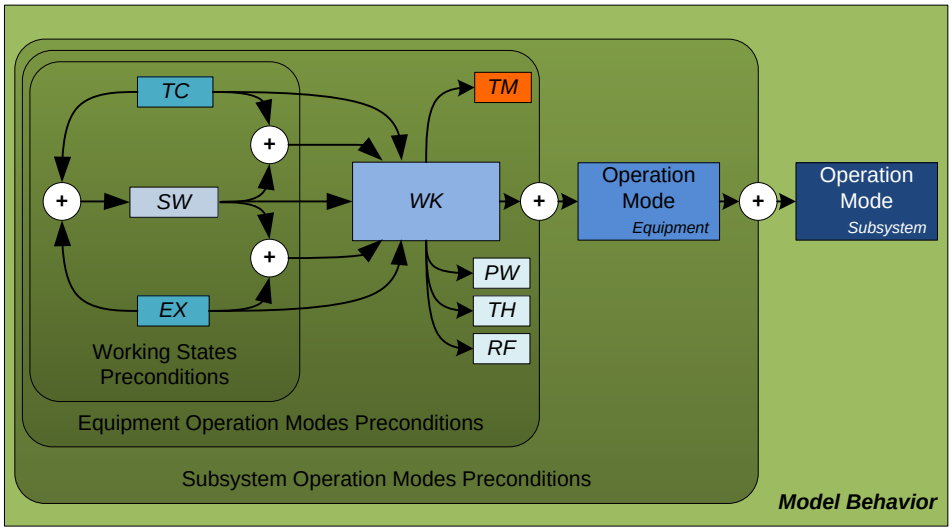

Fig. 8. Model Behavior Elements.

Figure 9 sums up all three sections into one diagram, showing the information data from the Design Documents as described by the Subsystem Engineers to be represented in a format that is both understandable and usable by the Software Engineer and her team, without prior knowledge of the subsystem. A format that is also clear enough for Subsystem Engineers, without prior knowledge of software representation, to validate the information contained within the model.

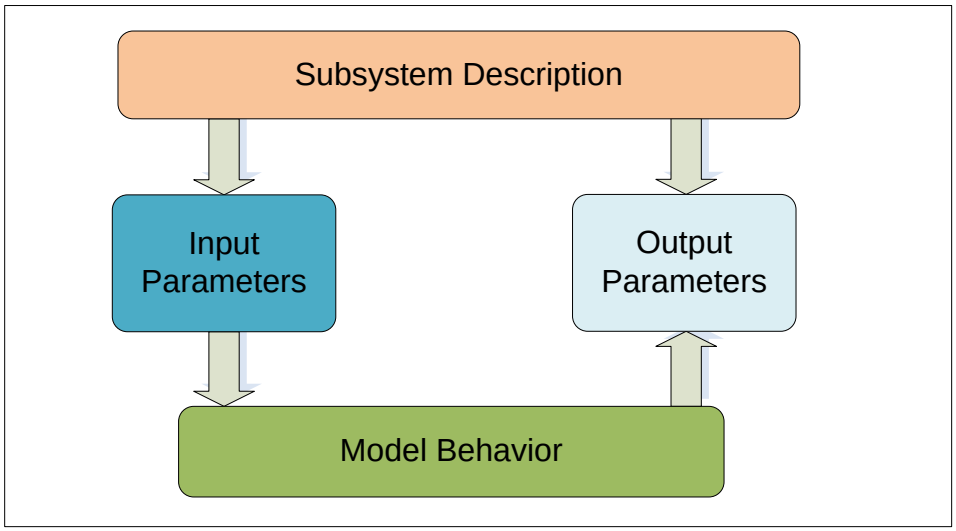

Fig. 9. Sections Interconnection.

CONCLUSION

Before the current Subsystem Model Specification documentation standard was approved, two earlier proposals to document the for the CBERS 3&4 satellites simulator were rejected by the Subsystem Engineers. The difficulty to achieve a consensus was mainly due to the different skills and backgrounds of the people involved. The subject of this paper could be rewritten as a short story about the glorious search for a common language understandable by electrical engineers, mechanical engineers, systems engineers, mathematicians, physicists, software designers and programmers.

The first attempt consisted on using algorithmic pseudo-codes to represent the knowledge. It was hailed as excellent to those familiar with programming languages, the software engineers and programmers. However, this approach had inherent difficulties concerning behavior rules validation. Subsystem Engineers reported trouble while checking them for correctness, especially when working with complex rules combining several precondition parameters at once. Deeply nested sequences of ifs were deemed impossible to verify by visual inspection. While some Subsystem Engineers proposed automated methods for validation – they actually suggested the use of software to validate software – others simply complained that they were not familiar with such notation and, therefore, would not even dare to try checking it.

The second attempt was based on a set of tables derived from the operational database of the Brazilian Satellite Control System. It had initially received high expectations, but soon its flaws became evident. The main reason for its failure was that the operation database lacked the structure required to represent all possible working states and power figures necessary for proper simulation.

The current solution is based on the third attempt and, so far, has been accepted by every people involved. Discussions held during the Review Meetings and the Review Reports issued thereafter have been instrumental to improve the modeling standard. In fact, problems in understanding how to model a subsystem once led Modelers to discuss a draft Subsystem Model Specification with the staff of the subsystem

manufacturer. After several meetings, the Subsystem Model Specification was not only approved, but the analyses and discussions led to the discovery of hardware issues that resulted in changes to improve its design.

Currently, roughly half of SIMC3 subsystem models must still be documented and approved, but the development team under the lead of the Software Engineer has already implemented some of their functionalities based on expertise acquired from previous programs. In the future, the use of the SYSML language to help documenting our models is under consideration. The SIMC3 development team is proudly aware that a work of utmost importance was completed, in creating a solid culture inside the institution concerning the writing of models for satellite simulators. We have learned that the difficulty in obtaining an easily understandable language to represent a complex system is not really felt until one actually tries to find one.

ACKNOWLEDGMENTS

The SIMC3 development team would like to thank all the Brazilian Subsystem Engineers and AIT Engineer, especially Guilherme Venticinque, currently working on the CBERS 3&4 satellites. Without their help, the Subsystem Model Specification standard could not have reached its current state.

REFERENCES

[1] Ambrosio, A. M.; Cardoso, P., E.; Orlando,V.; Bianchi-Neto, J. **Brazilian Satellite Simulators: Previous Solutions Trade-off and New Perspectives for the CBERS Program**. Proceedings of the 8th Conference on Space Operations (SpaceOps 2006), 19 Jun – 23 Jun. 2006, Rome, Italy. American Institute of Aeronautics and Astronautics - AIAA, 2006.

[2] Barreto, J.P.; Hoffmann, L.T.; Ambrosio, A.M. **Using SMP2 standard in operational and analytical simulators.** Proceedings of the 10 th Conference on Space Operations. 25-30 April 2010, Huntsville, Alabama, USA. **SpaceOps** 2010. American Institute of Aeronautics and Astronautics – AIAA, 2010.

[3] Rainer, L.B. – Space Modeling and Simulation – Roles and Applications throughout the System Life Cycle, 2004. Aerospace Press, USA.

[4] Tominaga, J; Ambrosio, A.M. RT-SRS-1022. CBERS3&4 Simulator Models Technical Specification Guidelines. INPE, 2012.