

Using SMP2 standard in operational and analytical simulators

Joaquim Pedro Barreto, Leandro Toss Hoffmann, and Ana Maria Ambrosio
National Institute for Space Research - INPE, São José dos Campos, Brazil, 12227-010
{joaquim, hoffmann, ana}@dss.inpe.br

Satellite simulators can be reused within a mission for different purposes and also reused among missions since there are generally several common points between one satellite to another. Consequently, the reusability is a fundamental requirement to be taken into account in the design of a satellite simulator project. The adoption of specific standards for satellite simulators can strongly enforces reutilization. The CBERS 3&4 satellite simulator, being developed at INPE, adopts object orientation, design patterns and the concept of software components in order to achieve a reasonable level of reusability. However no special standard is adopted. The Simulation Model Portability 2 (SMP2) standard established by European Space Agency (ESA) is being considered for future adoption. This paper presents a cost analysis to adapt the existing infrastructure of the CBERS3&4 satellite simulator to the SMP2 standard. The aim of this analysis is to show the practicability degree of translating the already existing architecture to another that, besides keeping the simulator in an operational scenario, could also open the possibility to improve it by adding new components with higher fidelity, turning it into an analytical simulator and taking advantages of using the SMP2 as basis. The development effort of component modifications and ease to integrate new software applications were taken into account in the costs calculation. The paper discusses the process to modify the simulator from one architecture to another; it discusses the drawbacks, the progress and results of the cost analysis performed in the last months.

I. Introduction

Satellite simulators are very important tools as they allow to foresee and analyze abnormal behavior to validate operational procedures, validate the satellite control center software and training satellite operators. Thus, letting those decisions be taken either related to satellite control or satellite operational planning. INPE (National Institute for Space Research) has been developing operational simulators since 1991¹. These simulators were intended to prepare operators at satellite control center to monitor and control satellites. Concerning the architecture of these simulators, it has been improved through the years, where the first one was strongly based on structured analysis. The latest one being developed, the SIMC3 (Satellite Simulator for CBERS 3), comprises object oriented programming, design patterns and software reuse techniques. This architecture also allowed the SIMC3 simulator to be extended to a version that could also be used for mission analysis. Two products have been planned to be delivered from SIMC3 project. The first one is a tool for operators training. It allows configuring simulation sessions including routine and contingency situations that the operator could face within their activities of monitoring and commanding a satellite. The second one is a tool for mission analysis that will permit to study the impact of operations out of the nominal range, to evaluate tendencies in satellite behavior during a long time. As an example, this tool will allow engineers to foresee the batteries behavior when a higher demand of energy is required due to an unplanned activity. Moreover, the feasibility of including a new ground station for data reception. Despite the flexibility of SIMC3 architecture, some restrictions were encountered to customize the simulator for reuse in several missions with its multiple satellites. These restrictions are mainly attached to the simulation infrastructure that has imposed a strong coupling between models and the control module. Besides that, simulation services are limited, considering, for example, time representation and process scheduling. The SMP2 (Simulation Model Portability, version 2) standard presents characteristics that could minimize those deficient points of the existing architecture. Its communication structure, interface standardization and component based support could help in breaking the existing dependency in SIMC3, making it easier to reuse this simulator

among several satellites and missions as well. In this sense, the reuse improves productivity and medium and long-term reliability.

This paper presents an initial study of applying the SMP2 standard in the existing SIMC3 architecture, taking into account difficulties and extra work that could be necessary for this adaptation.

The paper is organized as follows, section II describes the satellite simulator SIMC3; section III summarizes the main characteristics of the SMP2, section IV discusses limitations of SIMC3 concerning future reuses and compares the SIMC3 solution with the ESA SMP2 proposal. Section V concludes the article.

II. Satellite simulator SIMC3

The CBERS 3&4 satellite simulator is being developed at INPE since 2007. The architecture of this satellite simulator comprises three main modules: Graphical-User-Interface (GUI), Models and Controller. The design of this simulator is based on object-orientation, software components and design-patterns concepts. As much as possible, software reuse is applied. The existing SATCS³ Support component, which is a library of classes developed for the satellite monitoring and control system, is used here, in order to reduce cost and time of the development effort.

Figure 1 shows these main modules.

In this architecture, each satellite subsystem is translated into a model. The execution of all models is controlled by the Controller module. The Controller is responsible of creating and controlling the simulation environment, allowing its starting, stopping, pausing and resuming simulation runs. The Controller encapsulates the following set of services inherited from SATCS Support library: logger, timer and events. The SIMC3 simulator has two types of events: systems events (error, warning, exception, etc) and model events. The former is used by logger service whereas the later is used to update the GUI as soon as a model updates its parameter values.

The scheduler service was implemented in SIMC3 using the time service provided by the SATCS library. The interface class between GUI and the Controller (GUIControllerInt) was designed to provide access from the graphical interface to the simulation control and make these modules as independent as possible.

Concerning the design of the Models module, firstly each model represents the behavior of each satellite subsystem, secondly, an abstract class defines the common models functionality, and thirdly each particular characteristic of a subsystem behavior is included in a class representing one model. In this way, each model derives from the base class and extra functionalities are added to the specific class according to the needs for reproduce the satellite subsystem behavior it simulates.

Also, there is an interface class between the Controller and the Models (ControllerInt) to provide communication. Before starting the simulation, the Controller must set the session configuration parameters with their initial values. During the simulation, these values can be changed dynamically by an intervention of the simulation conductor.

The communication among models is provided via a blackboard mechanism implemented by the Controller. The CSIMAbstractParam class makes use of a class from SATCS Support library which supports a metadata processing for TM and TC parameters handling. This class provides, among others, methods for setting/getting parameters values. Then, a model can change the value of a parameter directly in the blackboard and another model will be able to read that value through the services of this class. Three types of parameters are specialized: satellite parameters, simulation parameters and telemetry parameters. This allows run-time parameter alterations occur in different ways according to the parameter type.

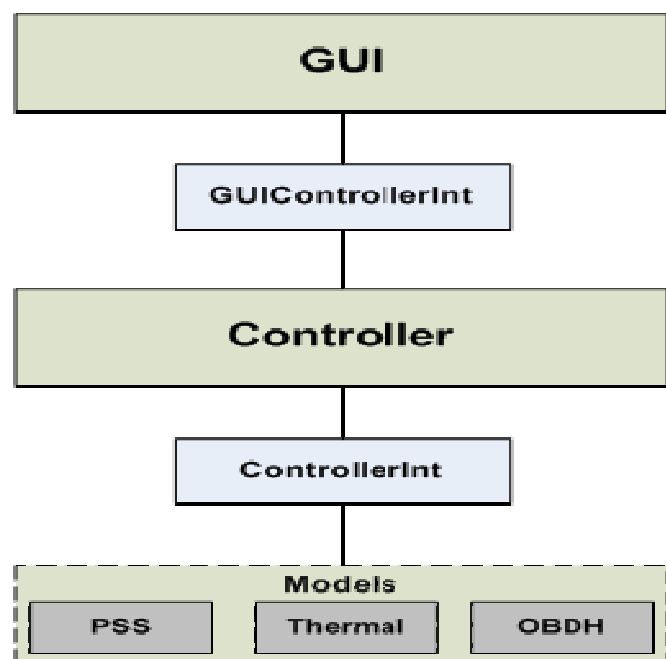


Figure 1. SIMC3 main architecture.

Figure 2 shows the classes implementing the blackboard mechanism.

A particular concept added to this simulator is the *fail handling*, whose purpose is to allow introducing unexpected and undesirable behavior in the satellite simulation. These misbehaviors can be introduced along with the simulation session, in predefined times or dynamically on simulation conductor's request. This facility helps to train operators to facing and react under an unexpected event or in disagreement with what was expected; for instance, when a key is not closed after a telecommand be sent to satellite or a payload is turned on after a telecommand is sent to turn it off.

Based on these characteristics a version of SIMC3 simulator can be easily fitted for operators training purpose. For mission analysis purposes, it is required that the models have high fidelity concerning orbit propagation, thermal, OBDDH, Power supply, TM/TC, etc.

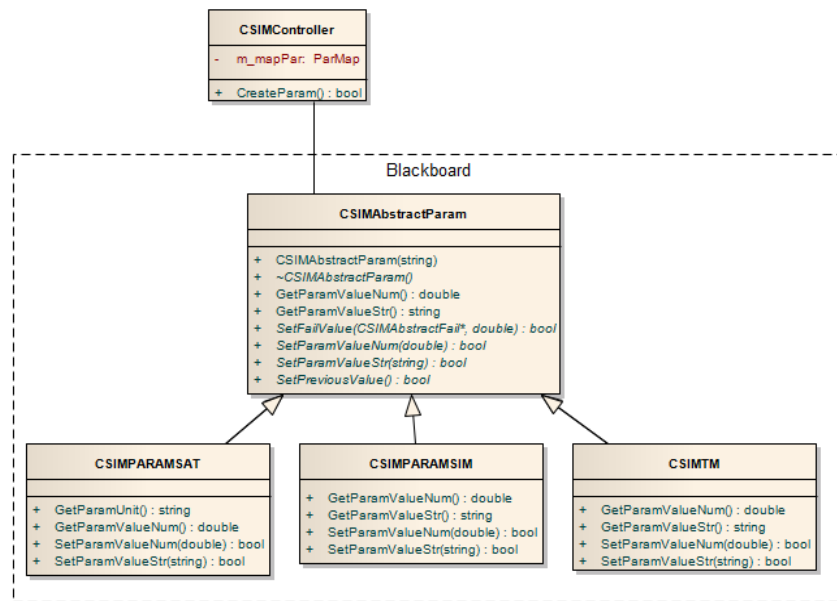


Figure 2. SIMC3 blackboard mechanism.

III. A standard for satellite simulation

The Simulation Model Portability 2 (SMP2) is a standard proposed by ESA² in 2005. The aim of this standard is to promote portability of models to be used in several simulation environments, different operational systems, consequently promoting a higher reuse of models. The SMP2 strategy of defining a standard of interfaces makes all satellite models, simulation services and simulation environment be developed according to this standard, enforcing portability, reuse and interchangeability. That allows different versions of the same model to be used for the same simulation set, without compatibility problems, as long as all of them were developed under the same interface structure. Moreover, each version of the same model can be turned into an independent component capable of being used in others simulation environments.

In a high level view, the SMP2 has three abstract layers:

- the physical level (the modeled engineering system)
- the simulation modeled system, and
- the software system which implements the simulation model.

The first level deals with the actual system being modeled.

The second layer represents a *Platform Independent Model*, whose purpose is to specify the models that will be simulated, but without any concern about how they will be developed and executed. In this phase, generic models already existing may be used as basis for generalization into specific models.

The third layer deals with *Platform Specific Model*, that is, the object code of the models to be executed in one specific computer architecture.

The SMP2 interacts directly with second and third layer, but, because it uses engineering data as input, also the first layer is affected. This view allows creating flexible simulation environments, based on models previously developed for engineering and applicable to process a product life cycle.

The architecture of SMP2 covers two types of components: the simulation model instances, which corresponds to the specific behavior, and the simulation environment which provides services.

Figure 3 illustrates the SMP2 architecture.

Four mandatory services should be provided by simulation environment: Time Keeper, Event Manager, Logger and Scheduler.

The Time Keeper service provides four kinds of time: a relative simulation time, a relative mission time, an absolute epoch time and the computer clock time.

The Event Manager service provides mechanism for global asynchronous events to be broadcasted.

The Logger service allows recording all messages generated either by models or by services.

The Scheduler service calls registered entry points based on cyclic or timed events and works together with Time Keeper service.

The communication is established via interfaces in two ways: model to simulation environment and model to model.

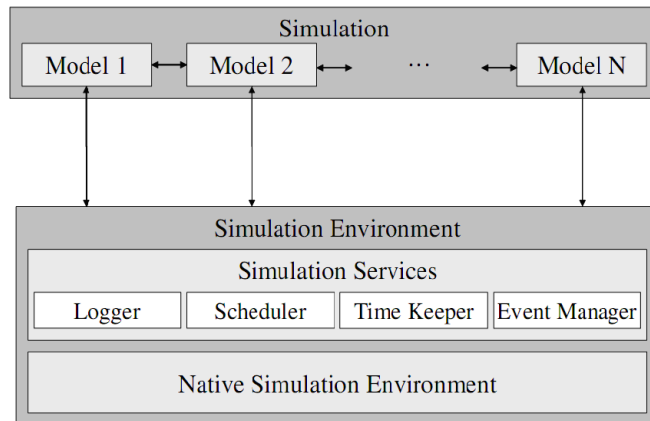


Figure 3. SMP2 architecture.

IV. Comparing the SIMC3 architecture and the SMP2 standard

This section discusses the study performed in order to make use of the best solutions proposed in SMP2 standard to the SIMC3 simulator without losing both the particular characteristics and all the good implemented solutions in the SIMC3..

The SIMC3 architecture provides a feasible solution, but with some limitations for reuse and software expansion purposes. The most limited point is related to communication between models and communication between the Models and the Controller modules. These communications are tight coupled, presenting some restrictions if modifications are required. Whenever a model needs to read/write data from/to another model, the reader and the writer should access the blackboard area supported by the Controller. This solution implies, for example, that all models know about other models internal parameters to access their values. The SMP2 inter-model communication could be the solution to eliminate the SIMC3 blackboard mechanism.

The expectation is that from the analysis performed in our study, it will be possible to figure how much effort would be necessary to make the SIMC3 comply with the SMP2 standard, which are the parts of the SIMC3 requiring more modifications, etc...

Several analysis and comparisons were performed, whose results are presented as follows. A first comparison took into account the services provided by both architecture and the number of classes involved. The cost of software modification based on statistics methods was also performed. At the end a discussion about software component advantages is presented.

A. Comparing SIMC3 simulator and SMP2 standard services

Both SIMC3 and SMP2 provide services to support models execution. In this aspect, the purpose is to verify if there is any mandatory service which is required by one architecture and not provided by the other. Table 1 shows the SIMC3 simulator services whereas Table 2 presents the SMP2 mandatory services.

Table 1. SIMC3 services.

Service	Description
Scheduler	Controls the execution of scheduled or cyclic events
Time	Controls the simulation time and provides support to Scheduler service
Log	Stores the simulation event messages in a historic database
Events	Notify graphical user interface about model events triggered during the simulation
Fail	Schedule and change parameters values to be triggered during the simulation session

Table 2. SMP2 mandatory services.

Service	Description
Scheduler	Provides entry points calls based on timed or cyclic events
Time	Provides four different kinds of time to the simulation: absolute time, relative simulation time, relative mission time and Zulu time (computer clock)
Log	Stores event messages (error, warning, event and information)
Events	Provides mechanisms for global asynchronous events management

Practically all services currently implemented on SIMC3 are specified in SMP2. However, SIMC3 does not implement all the service resources that SMP2 recommends like simulation control time or mission control time. And even when implemented, some resources are not easily customizable.

The scheduler service of the SIMC3 has a different structure when compared to SMP2 IScheduler interface, since its purpose was to schedule the execution of fails, that is, to change the values of parameters that represent the fail, at specific times. This service is not used for scheduling models. Models communicate with each other through the blackboard mechanism. For each simulation step, all models execute their inner code and write/read data from/to the parameters created at blackboard, according to their needs.

The *event manager service* has the most different structure in both architectures. The SIMC3 events service operates through a message system mechanism (first Windows message, later Qt signal/slot). Nowadays, this event message implementation is used to update the graphical user interface as soon as a parameter value is changed by any model so it can be visualized by a simulation conductor. It is not used for models communications at the moment in order to avoid a processing overhead. On the other hand, SMP2 defines a provider-consumer architecture where the consumer of and event must register in order to be notified as soon as this event happens. The SMP2 solution for event management is more complex than that designed for SIMC3.

The *fail service* of the SIMC3 is mostly to be used for operators training. It consists of introducing, at predefined times, unexpected behaviors in order to train operators reaction. The simulation should therefore controls the instant the anomaly triggers, which model it affects and how long it will take. Once the concept of fail is directly related to the affected models, it should be verified whether it remains a service or could be integrated to the models and then to have communication with the simulation control when necessary.

It is important to notice that SMP2 includes the facility to create user defined services. Then, if SIMC3 fail concept remains a service, it could be added to a SMP2 service implementation.

Even though SIMC3 implementation is a little simpler and the comparison was superficially in terms of **services contents** this first comparison showed that there is a high similarity in both set of services.

B. Comparing number of service classes

Another way to assess the effort to adapt SIMC3 to SMP2 is to count the number of service classes from both architectures. This can give us an idea whether there could be an increment or a decrement in the number of classes. Although it could not be conclusive about the amount of work involved in a likely adaptation to be done, it could anticipate numbers to help taking future decisions.

Table 3 shows the minimum number of service classes in SIMC3 and SMP2 that is involved in this comparison. This number includes the SMP2 interface classes for the services shown in Table2 but, does not take into account the classes of the SIMC3 that manage fails.

Table 3. Summary of service classes.

	Total of Classes
SIMC3	18
SMP2	22

It was considered only one implementation-class for each service interface defined in SMP2. However, there should be an increase since it is not possible to see in advance any eventual modification or specialization required during the implementation, either for needs or for convenience.

Considering that SIMC3 has a total number of 72 classes, it is estimated 25% of the total classes to be rewritten in order to have service classes adapted to SMP2. This number includes classes from models, blackboard and services, but does not include classes from graphical user interface (GUI).

Next, it is shown a numerical analysis using a software size calculation technique to precise the cost of such modification.

C. Cost to adapt the event service from SIMC3 to SMP2

To evaluate the cost of applying the SMP2 standard to SIMC3 the *events service* it was chosen as a case study. This service presents the major differences in the solution implemented in the SIMC3 when comparing to that proposed by the standard. The SMP2 structures the events services throughout five interfaces: IEventSource, IEventSink, IEventProvider, IEventConsumer and IEventManager. An event source (provider) can be connected by a consumer (if it holds an event sink of the same type). Whenever an event is triggered at the provider, all consumers are invoked.

Figure 4 shows the structure of event service in SMP2.

In SIMC3 the event solution operates with messages, having one abstract class(CSIMAbstractEvent) and several derived from this base class, overriding the Notify method, which triggers the specific message and therefore the event attached to it, as shown in Figure 5.

As previously stated, the event service in SIMC3 is used only to update the user interface. As soon as an event occurs, the message is sent, the GUI is notified and updates the information, like eclipse, visibility and others, in the screen. Event messages are not used for models communication since it is done via the blackboard mechanism.

The metric of size, taking into account the physical SLOC (Source Lines of Code) was used to evaluate the cost to modify the SIMC3 event service. The technique applied is detailed in the Handbook for Software Cost Estimation⁴. It states that the approach to estimate the sizes could use analytical and statistical steps, both having SLOC as basis. The first method requires a similar software developed previously, in order to have an analogy based estimation. The second method involves developing estimates based on expert judgment and statistical calculation.

The event service adaptation study showed that the SMP2 IEventSink interface is very close to CSIMAbstractEvent, since their functionality relies on the Notify function. Then it is an indication that the comparison between them could be done based on the similarity.

Table 4 shows the comparison using the first method of the technique pointed out before (similarity).

The mean value on the following tables is calculated according to Program Evaluation and Review Technique (PERT) as follows:

$$\text{Mean} = (\text{Least} + 4 * \text{Likely} + \text{Most})/6 \quad (1)$$

Table 4. Similarity based Estimation.

Segment	Analogy Size	Estimate 1			
		Least	Likely	Most	Basis of estimate
New IEventNotify	306	250	306	500	IEventSink similar and same size as CSIMAbstractEvent

The likely value (306) is the value of SLOC found in SIMAbstractEvent and all of its derived classes.

For comparing the others SMP2 interfaces, a totally new class design and code has to be considered, since SIMC3 has no class with similar functionalities to be compared with SMP2. So, the second method (statistically) is more indicated to make this estimate. Table 5 shows the numbers found.

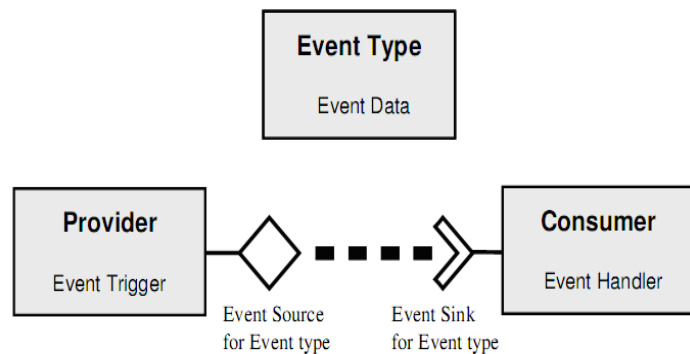


Figure 4. SMP2 Event-based design.

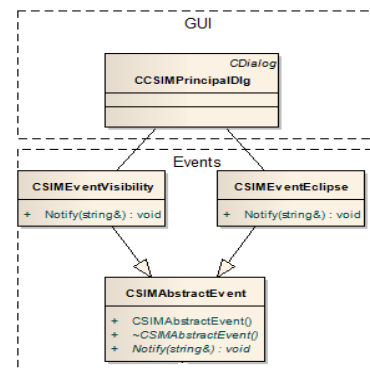


Figure 5. SIMC3 Events design.

Table 5. Statistic based Estimation.

	Estimate 2			
	IEventSource	IEventProvider	IEventConsumer	IEventManager
Least possible size (a)	50	50	50	150
Likely size (b)	60	60	60	200
Most possible size (c)	100	100	100	300
Statistical ($S = (a + 4b + c)/6$)	65	65	65	208

Table 6 summarizes the mean size estimates, giving the total size estimated.

Table 6. Total size estimate.

Segment	Mean size
Estimate 1 (IEventSink)	329
Estimate 2 (IEventSource, IEventProvider, IEventConsumer, IEventManager)	403
Total size	732

The total size estimate indicates that it could be necessary, at least, two times more of SLOC than the quantity already existing in SIMC3 events to implement all SMP2 interfaces for events service.

Concerning the number of classes, the *minimum* number expected to implement the SMP2 event interfaces would be half of the existing in SIMC3 event classes. This higher number in SIMC3 is because it is necessary to create a new class derived from the abstract event class to treat every new event.

D. Models Adaptation

Concerning the models adaptation from SIMC3 to SMP2 standard, little work is required..

In fact, if all services of the SIMC3 were translated to a SMP2 framework, as well as the control to an ISimulation interface implementation, the models would be the last items to be converted.

As the models represent a well defined behavior of a satellite subsystem, their internal content could not be affected by the interface requirements. The major part of the work would be to make the existing models compatible with SMP2 in order to perform the steps required for setting up the simulation. In this setup phase the models have to be configured and may perform some other tasks related to services and other models like, publish fields and operations or make some custom initialization.

Again, the hardest part goes to the models intercommunication solution, allowing all models and services to communicate themselves.

Even if it is considered that all models of the SIMC3 would remain in one class per model, it would take one extra class for every model to integrate the existing model and the SMP2 interfaces. This extra class could encapsulate all the work needed to provide the functionalities described above, so the models would behave correctly inside an SMP2 simulation environment. Perhaps some specific details of a model could require any extra modification in any model.

E. Component-based technology versus models

One of the advantages of adapting SIMC3 project to SMP2 is the use of component-based technology, supported by the standard, to promote model reuse.

The interface standardization offered by components allows them to be reused in several versions of the same software or that several versions of the component being developed. These different versions of the same component can be changed during development or execution of a project, without causing any problem since their interface (input/output) is the same.

As SIMC3 for mission analysis is being developed by adding models one-by-one gradually towards a high fidelity simulator. The adaptation of the models into a software component may offer an easier way to interchange them among several versions of the same model. In this way, there could be several orbit propagator versions thermal model with higher or lower complexity (depending on the simulation purpose), a power supply model with different configurations of solar panel or batteries, etc.

This approach could allow that simulators for different stages or desired functionalities (training or analysis) be customized more easily, as well as different configurations of the simulator to different satellites.

V. Conclusion

The proposed architecture for SIMC3 satellite Simulator has proved to be a good solution for a dedicated satellite, when it comes to an operational scenario. Its breakdown into three modules separating user interface, simulation domain and models allowed a more controllable management of the software development.

Moreover, this architecture breakdown made it possible to improve the simulator by extending its functionality to become a simulator for mission analysis. This could help satellite experts to explore the use of satellite resources, in order to reach the limits of their characteristics. The key for this major step is the possibility of switching low fidelity models for higher fidelity ones.

Although the architecture has several positive characteristics, there are some limitations that could be noticed after the development of the first version. The main drawbacks are the inter-model communication and the tight coupling among models, simulation environment and simulation services.

In order to mitigate these limitations, the SMP2 standard was taken into consideration. Since its purposes are on the same direction as SIMC3 requirements, an analysis to verify the possibility of using the standard together with SIMC3 has started.

Although the simulation services in both architectures are quite similar, the SIMC3 service implementation is not as complete as SMP2 proposes. Events service should be almost entirely developed. Concerning the models, the main problem is to develop an inter-model communication to eliminate the blackboard mechanism existing in SIMC3.

Then, future studies should include an analysis of the simulation environment including models communication, addressing a simulation framework to support models. Despite the documentation made available by ESA about SMP2, it is not completely clear the amount of work necessary to implement SMP2 communication structure.

As far as this comparison could show, an adaptation of the SIMC3 to the SMP2 standard should start by making the service environment available to the simulation environment. Then, the services whose contents are somehow similar to log, time and scheduler should be adapted. Events should be the next one, as it would require completely redesign SIMC3 event service to comply with SMP2. The communication should come in the sequence. And finally, the existing models should be adapted to SMP2 models interface.

Acknowledgments

We would like to thank our DSS colleagues, particularly Simone Léo, for the discussions that allowed improving the SIMC3 description.

References

- ¹Ambrosio, A. M.; Cardoso, P. E.; Orlando, V.; Neto, J. B. "Brazilian Satellite Simulators: Previous Solutions Trade-off and New Perspectives for the CBERS Program," *Spaceops Conference, AIAA*, Rome, Italy, 2006
- ²SMP 2.0 Handbook, EGOS-SIM-GEN-TN-0099, Issue 1.1, 28-Feb-2005
- ³Cardoso, P. E.; Barreto, J. P.; Cardoso, L. S.; Hoffmann, L. T. "Using Design Patterns, Components and Metadata to design the Command and Monitoring Frameworks of the INPE's Satellite Control System" *Spaceops Conference, AIAA*, Heidelberg, Germany, 2008
- ⁴Kuykendall, F., (ed.), *Handbook for Software Cost Estimation*, JPL D-26303, Rev. 0, JPL, 2003.