A multithreaded rule-based satellite simulator for satellite control monitoring

Jun Tominaga[1], Mauricio Gonçalves Vieira Ferreira[2]
Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, São Paulo, 12227-010, Brazil

This paper describes a new software application intended for assisting human operators in monitoring the health of spacecraft in orbit. Its basic design is inherited from another satellite simulator developed for planning validation, which consisted of a single rule-based inference engine with associated database files. These database files contained all the system dynamics rules, internal state parameters, and scheduled events covering a whole simulation session. Despite its advantages, this approach results in complex systems that become harder to maintain as the number of rules and parameters increases. The new approach tries to address this problem by dividing the database into smaller modules, each comprising an inference engine with smaller database files. Several inference machines are run simultaneously as threads controlled by a manager module. All rule-based threads communicate with each other via a central data structure, maintained by the manager module, which stores all internal parameters of the simulator model. The simulation parameters values are represented internally in engineering units, which are encoded into telemetry data or decoded from telecommand messages by network interfacing modules, also running as threads. Simulated parameter values are compared to satellite telemetry received by ground stations in real-time during passes and, if the difference between corresponding values surpasses its predefined range, an alarm is triggered to warn the operator. The description of the simulator presented in this work comprises the satellite ground control system architecture, the rule-based inference engine and its database data structures, and the integrated architecture including the satellite control monitoring system.

I. Introduction

Telemetry, tracking and commanding (TT&C) activities for satellites built and maintained by the Brazilian National Institute for Space Research (INPE) are under the responsibility of the Satellite Tracking and Control Center (CRC-INPE). CRC-INPE consists of a satellite control center located at its headquarters in São José dos Campos, a primary TT&C ground station at Cuiabá, and a secondary TT&C ground station at Alcântara. The locations of these facilities are indicated in the following map (Figure 1) as SJC, CBA and ALC, respectively. The control center and the ground stations are interconnected through a private ground link.

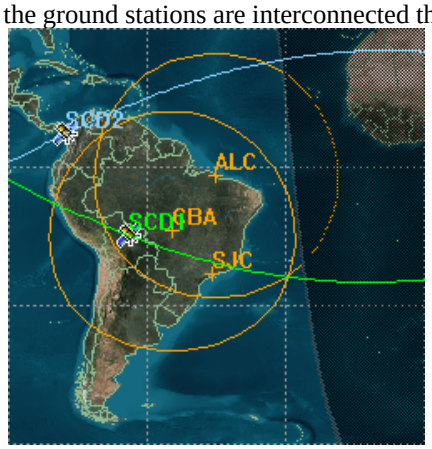

Figure 1. CRC-INPE Satellite Ground Control Facilities

[1] Engineer/Technologist, CRC (Satellite Tracking and Control Center), Av. dos Astronautas, 1758, prédio CCS, São José dos Campos, 12227-010, Brazil, jun@ccs.inpe.br
[2] Doctor Engineer/Researcher, CRC (Satellite Tracking and Control Center), Av. dos Astronautas, 1758, prédio CCS, São José dos Campos, 12227-010, Brazil, mauricio@ccs.inpe.br

In order to perform satellite mission operations around the clock, CRC-INPE is staffed with satellites control operations personnel working in four continuous shifts of six hours each. Under normal circumstances, human operators are always manning each of the facilities of CRC-INPE in pairs, two at the satellite control center, and two at each of the TT&C ground stations. Although the number of dedicated staff assigned to this task seems high at first glance, and indeed the work load is very light under normal circumstances, for the last two decades CRC-INPE has sought to maintain this status-quo for operational reasons.

In case of emergency, these satellites control operators are instructed to execute contingency procedures according to manuals and procedural instructions. Usually under such circumstances, for example when the TT&C data network goes down against all odds, or satellite equipment fails to respond properly to telecommands, one operator becomes responsible for processing the contingency, while the other one executes routine tasks that remain unaffected by the problem. For instance, if a dual data link failure between the satellite control center and the TT&C ground station occurs, one of the ground station operators becomes responsible for contacting the network service provider, while the other one follows the flight operations plan as scheduled, effectively taking on the role of the control center. If an onboard failure is detected, one of the control center operator calls the satellite systems engineer for notification and instructions regarding the faulty spacecraft, while the other one takes care of the remaining satellites as usual. However, if a contingency occurs while one of the operators is out of work for any reason, the amount of tasks assigned to the remaining operator can easily become overwhelming.

Because of the ongoing trend advocating cost reduction, CRC-INPE has found that justifying the necessity of hiring new operators has become increasingly difficult. Therefore, one of the steps under implementation at CRC-INPE to counter this problem is automating some of the processes that constitute the satellite ground control system.

II. The Satellite Ground Control System Architecture

The overall architecture of the satellite ground control system under implementation at CRC-INPE is shown in the following diagram (Figure 2).

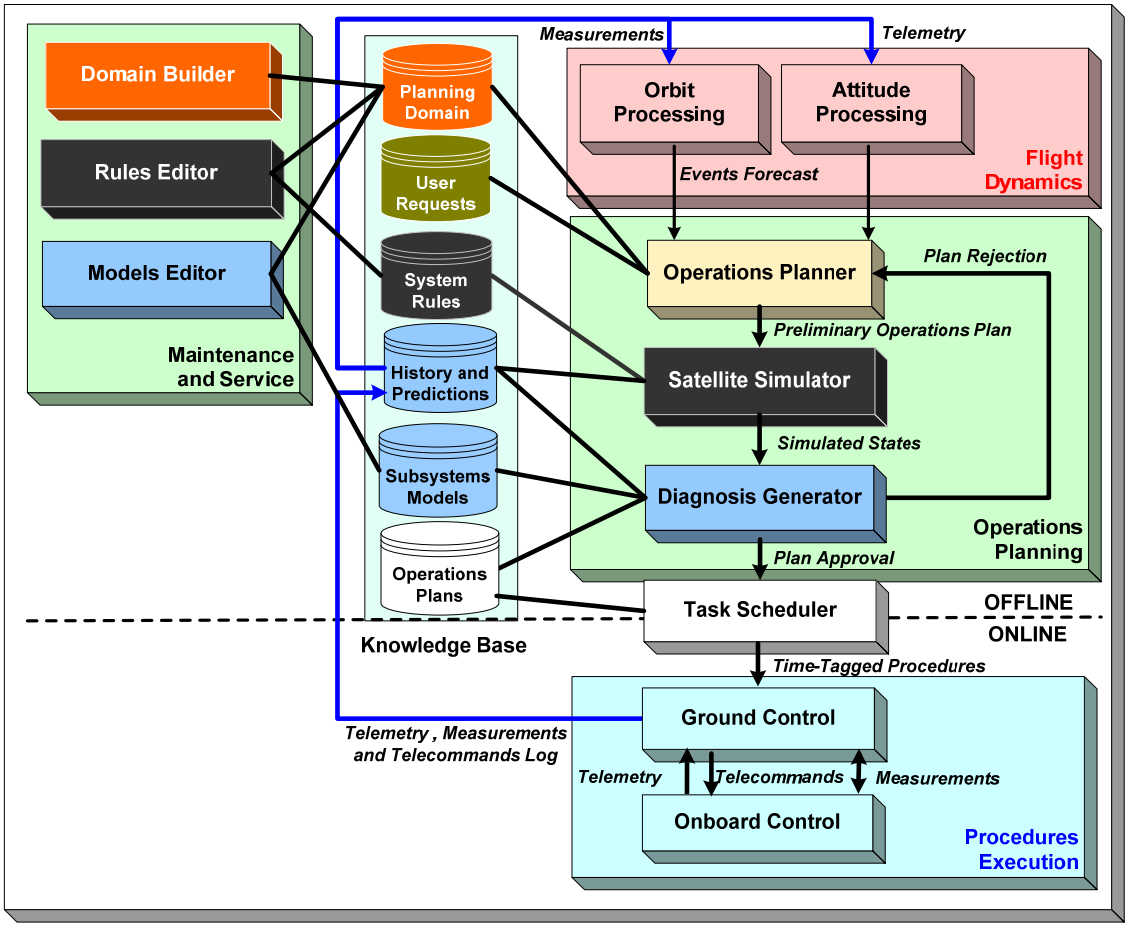

Figure 2. Satellite Ground Control System Architecture

The satellite control center is responsible for coordinating the activities of TT&C ground stations in order to acquire, store and transfer satellites mission data, for further processing and distribution by mission data centers. To achieve this, real-time procedures are executed when the satellites fly over the footprints of the TT&C ground stations antennas. These tasks are preplanned based on satellite contact predictions generated by the flight dynamics team, in the satellite control center.

The following illustration (Figure 3) details the architecture of the procedures execution subsystem. The ground control has three main functionalities that consist of telemetry monitoring, telecommand transmission, and ranging and range-rate measurements. At every spacecraft pass tracked by CRC-INPE, satellite control center and TT&C ground stations operators are expected to be able to perform all those three procedures simultaneously.

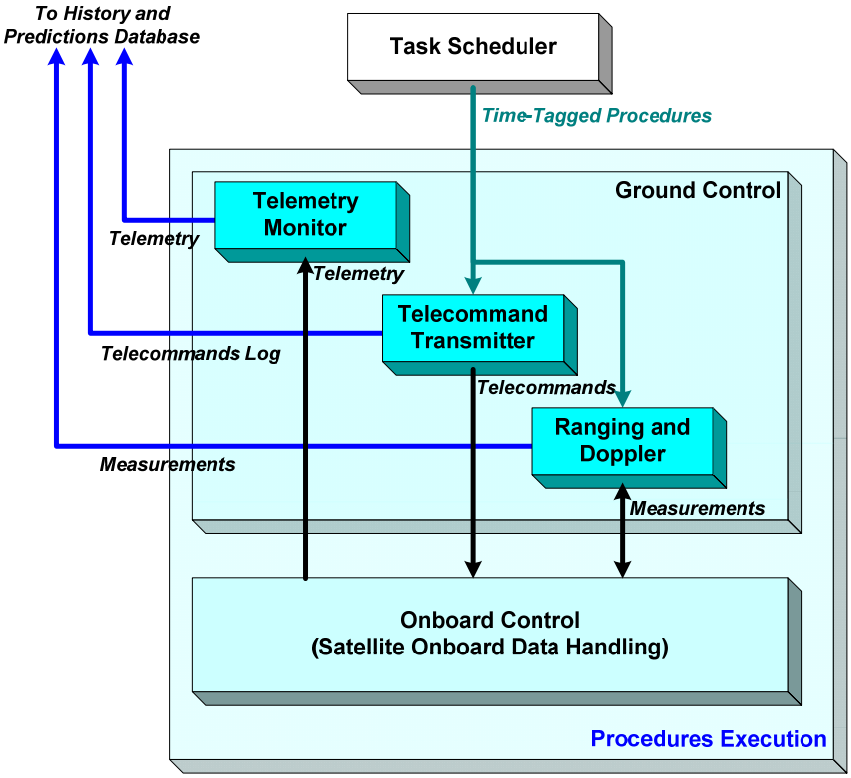

Figure 3. Procedures Execution Subsystem Architecture

During the routine operations phase, telecommands and measurements are executed as scheduled by the operations planning. A task scheduler can therefore use this information and program each of the procedures in a time-tagged queue for automated execution. Telemetry monitoring is performed for the purpose of verifying the onboard reception of transmitted telecommands, as well as for overall satellite health assessment. If an onboard failure is detected, the task scheduler must be stopped by the operator. The contingency plan must be then executed manually. It must be noted that, under normal circumstances, the manual execution of real-time procedures is a very easy task for a skilled operator. However, the manual execution of procedures does have associated risks, especially on the hands of somewhat inexperienced operators who are alone and, even worse, under stress because of an ongoing contingency. A situation that should be avoided is one in which an onboard failure goes unnoticed because of another emergency. The most catastrophic result could be the total loss of a mission, caused by inadequate execution of routine operational procedures on a faulty satellite, which should have been placed in safe mode and left undisturbed until the problem is diagnosed.

Furthermore, although the telemetry monitoring software is conceived to issue warnings for out-of-range telemetry, some onboard failures are known to be trickier than others. For example, a piece of equipment comprising several units might have several telemetry thermistors, each capable of indicating the temperature of one of the component units, and a single status telemetry indicating the ON/OFF status for the whole set. In addition to this, the manufacturer would typically supply the operational temperature range for the equipment as a whole, instead of discriminating the expected range for each component part according to its working state.

In a monitoring system conditioned to display alarms based on telemetry statuses and thresholds, a failure which causes one of the component units to remain powered on after the execution of a telecommand to turn off all units could be very hard to detect. With the status telemetry indicating the OFF state and thermal telemetry increasing but still within the maximum allowed value, the telemetry monitoring software would assume that the telecommand worked without problems.

This type of problem would be detected if a satellite simulator supplied a list of expected values for telemetry parameters. It would indicate that all temperatures should be decreasing after a successful turn off, instead of increasing. Thus, to reduce the burden on the shoulders of satellite control operators and minimize operational risks, a proposed addition to the procedures execution subsystem includes an automated tool for performing preliminary health diagnosis and contingency alert. This could be implemented by adding a satellite simulator to the ground control software and a tool for comparing the satellite telemetry with simulated states, in real-time.

III. The New Procedures Execution Architecture

The following diagram (Figure 4) shows the architecture of the procedures execution subsystem including the newly proposed feature. The main idea behind this is to run the satellite ground control software simultaneously with a satellite simulator that reflects the satellite operating under normal conditions. If the real-time comparison between the satellite telemetry and the simulated parameters shows big divergences, it is supposed to be caused by an abnormal behavior on the part of the satellite. The comparative analyzer would then issue a contingency alert to warn the human operator. The automated execution of time-tagged procedures would be stopped, and the telecommand transmission forced into manual mode.

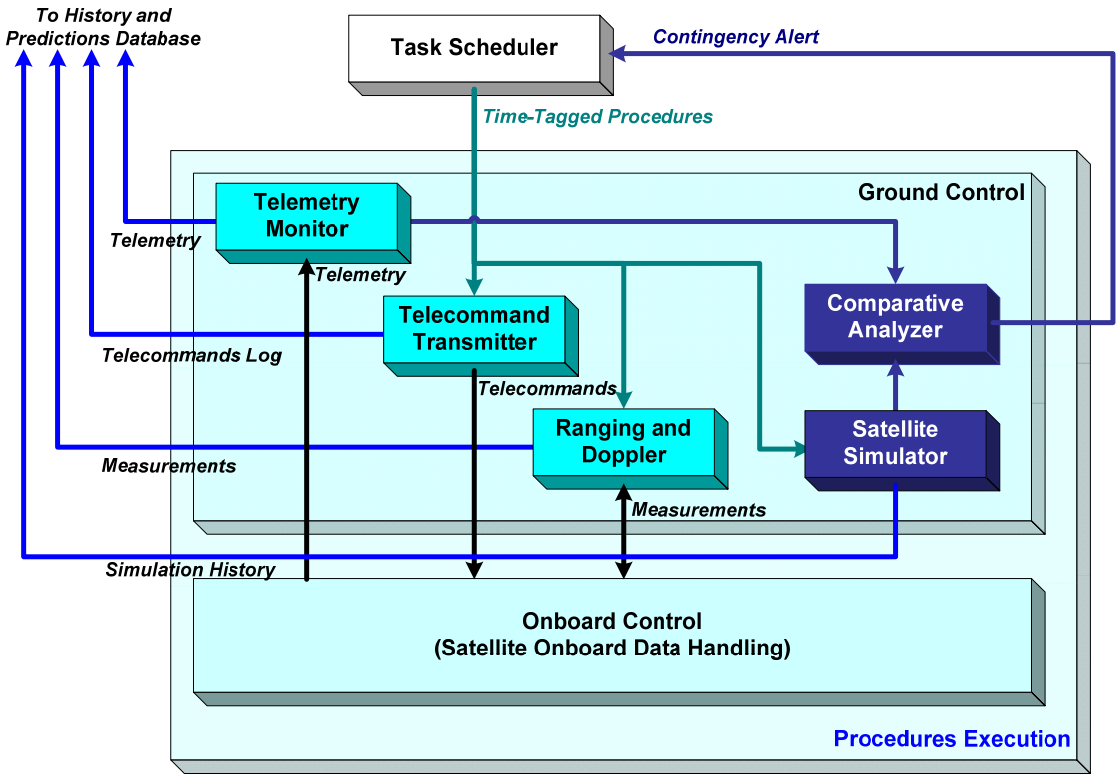

Figure 4. Newly Proposed Procedures Execution Subsystem Architecture

In this new architecture, the time-tagged procedures list is still managed by the task scheduler. The lists of telecommands to be transmitted, as well as ranging and range-rate measurements to be performed, are queued for execution prior to each predicted satellites pass as in the previous architecture. But now, this information is shared with a satellite simulator as well. The simulator is kept running alongside the satellite ground control software as a standalone application. In routine operations, the list of scheduled procedures suffices to ensure that the satellite and the simulator are well synchronized, since the transmission of telecommands is expected to follow the scheduled plan. That is, provided that the simulator internal state configuration was correctly initialized according to the operating mode expected from the actual satellite. The comparative analyzer retrieves simulation parameter values via network connection, and decoded telemetry from the telemetry monitor application through a special service requested by assembly, integration and tests (AIT) experts.

The time-tagged procedures used by the procedures execution subsystem are generated and supplied by the operations planning subsystem. It should be noted that the operations planning subsystem includes another satellite simulator for the purpose of validating flight operations plans, as shown previously in the overall satellite ground control system architecture (Figure 2). In order to cut development costs, an attempt was made to reuse as much as possible of the planning validation simulator in the new procedures execution subsystem.

IV. The Satellite Simulator for Planning Validation

In the operations planning subsystem, the satellite simulator for flight operations plans validation checks the effects of scheduled telecommands on the satellite before they are handed over to the procedures execution team. If diagnosis of the simulation results indicates an unsafe internal state, planning errors will be evidenced, and then corrected, before the procedure tasks are scheduled for execution. Because this simulator was designed specifically for planning verification, it has no interfaces allowing it to be used in real-time simulations. Its interfaces are evidenced in the planning subsystem design architecture presented below (Figure 5).

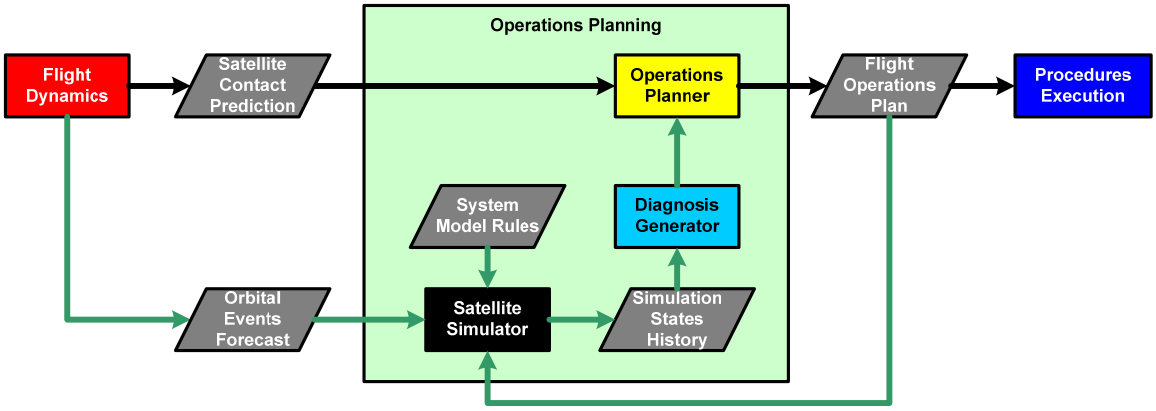

Figure 5. Operations Planning Subsystem Architecture

The data interfaces of the simulator are illustrated below (Figure 6). The simulator application, shown in black, consists basically of a rule-based inference engine. The associated knowledge base files contain the system model rules, which describe the dynamic behavior of internal state parameters and how they change according to triggering events, as well as the simulated states history, from where the initial state is obtained. A pre-generated schedule of rule-triggering events is obtained from two different sources. Planned actions and associated telecommands are obtained from the flight operations plan under test, whereas environmental events are supplied by the flight dynamics orbital forecast.

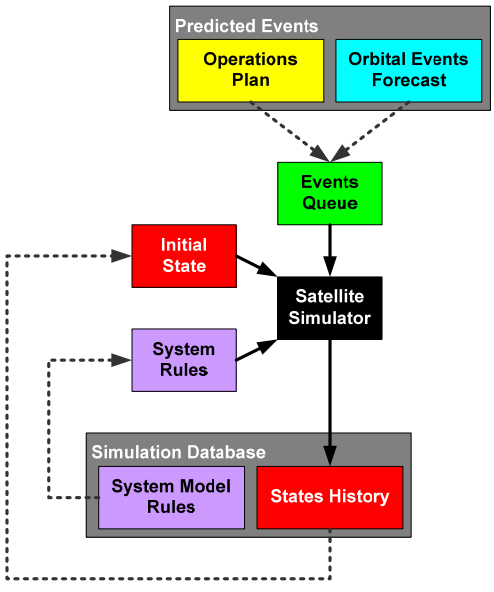

Figure 6. Planning Verification Satellite Simulator Architecture

The next figure (Figure 7) shows the execution phases of the satellite simulator and the associated data structures processed at each one.

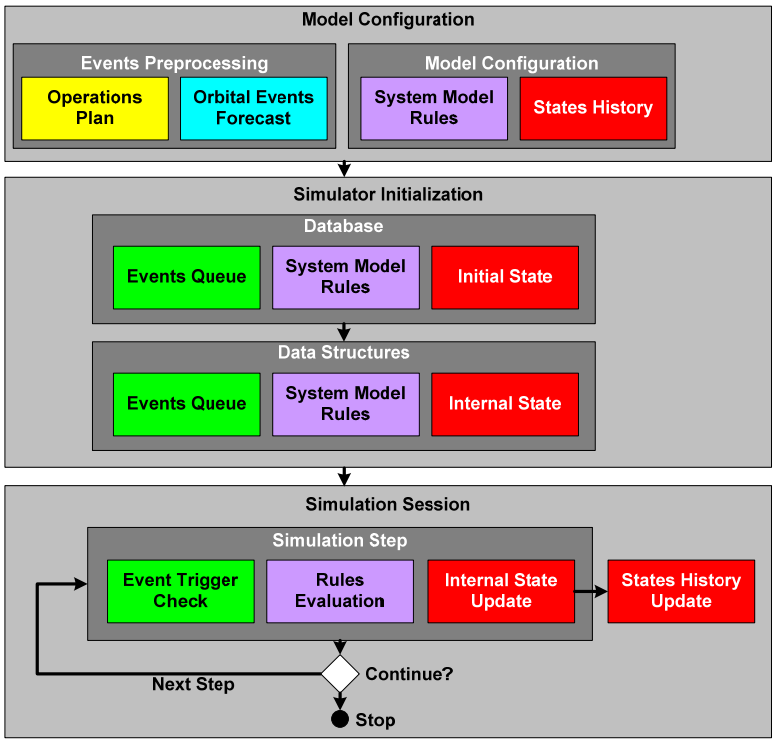

Figure 7. Execution Phases of the Satellite Simulator for Planning Verification

Before the simulator is set to run, it must be properly configured. The operations plan containing the scheduled procedures and the orbital events forecast are preprocessed to form an events queue data structure. This structure contains a list of time-tagged event parameters that are expected to be triggered during the simulation session. In addition to this, system model rules must be up to date. Rules for routine operations in degraded mode differ from those designed for the beginning of life, therefore on must take care to assure that the database reflects the satellite behavior and the operational restrictions correctly. At last, the simulated states history database must be available, or otherwise the initial state of the simulated model must be explicitly configured. All satellite simulator data configuration is performed by means of external tools.

Then, as the simulator execution starts, the simulated system information is loaded from database files to memory data structures. The events queue, system model rules and the internal state of the simulated satellite at the start of the simulation are set at this phase.

Following this, the simulation session starts. At the start of each simulation step, event parameter values are set based on the comparison between the simulation time and elements present in the events queue. The whole set of system model rules are then evaluated sequentially to determine the internal state at the next step. The updated internal state is exported to a simulated states history database, and the stop condition is evaluated to decide whether the simulation continues or halts.

One of the main factors that led to the choice of this simulator software architecture was the necessity of a highly reconfigurable model to represent the internal dynamics of the simulated system. Older satellite simulators developed by INPE used databases that allowed some simulation parameter values to be modified, making them reconfigurable to some extent. However, the rule-based architecture allows the storage of full expressions, as well as their triggering conditions, inside corresponding knowledge base files. This feature was considered to be of extreme importance, since experience had already taught CRC-INPE that onboard failures do come in bizarre combinations that result in unpredicted operating modes. Safe operation procedures that were once valid may thus become invalid, as old and degraded equipment and components may no longer be able to withstand such use. Thus, simulation models should be changed from time to time to reflect the current behavior of the spacecraft. The concept behind the rule-based modeling was to make the updating process of the knowledge base, ideally to be performed by satellite design experts themselves, the least painful possible. This led to the development of database structures designed to reflect the knowledge of these experts.

The scope of the planning validator simulator was limited to a small subset of telecommands intended for use in routine operations and internal parameters affected by them. This approach reduces the amount of telecommand and telemetry parameters to be included in the system model rules. However, for the purpose of real-time monitoring, the scope must be widened to include the entire set of telecommand and telemetry defined for the satellite. The amount of parameters and rules was expected to increase immensely, and experts responsible for validating the implemented model started to point out that its complexity would extend beyond their knowledge. In order to address this problem, the suggested approach was to divide the system model into smaller, subsystem models, in the same way the actual satellite development program is managed.

V. The Satellite Simulator for Real-Time Monitoring

The general architecture of the satellite simulator conceived for use in the procedures execution subsystem for real-time monitoring is shown in the following diagram (Figure 8). Basically, there are subsystem modules responsible for the simulation properly and interface modules for information management.

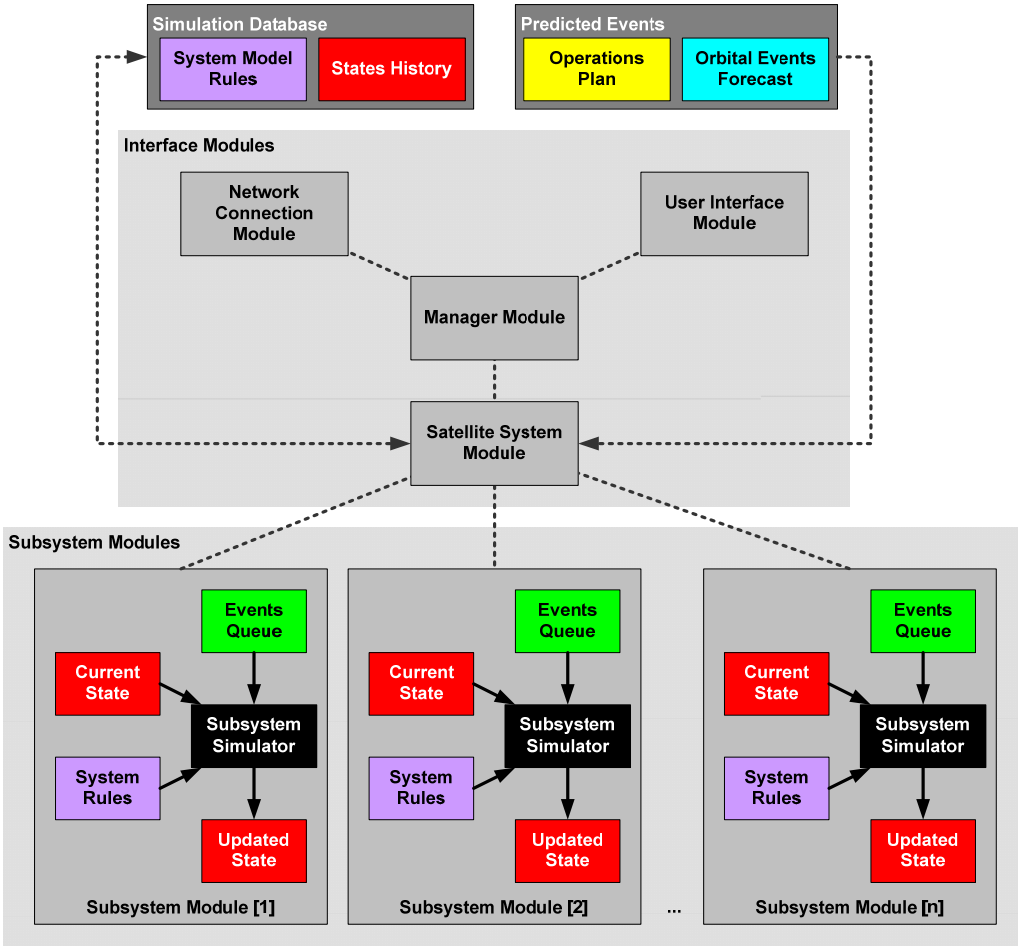

Figure 8. Planning Verification Satellite Simulator Architecture

One can see that in this simulator, the subsystem modules correspond roughly to the planning verification simulator shown previously (Figure 6). The main difference is that they no longer interface with the simulation knowledge base. Instead, each of them relies on the satellite system module for its initialization. In the simulation session phase, the current state supplied by the satellite system module is updated according to the evaluation of each subsystem events queue and subsystem system rules. The updated subsystem state is then retrieved by the satellite system module.

The satellite system module is responsible for managing the interfaces of all subsystem modules. It retrieves the system information from the simulation database and event predictions, and distributes them among the subsystem modules for initialization. During the simulation session, this module makes the simulated state available to other information modules and performs the update of the simulation states history.

The thread manager module, also known as the main module, communicates with the satellite system module, the user interface module, and the network connection module. Its main purpose is to initialize the other modules and facilitate the data exchange between them.

The user interface module provides the means for a human operator to monitor the internal state of the simulator. State parameters are displayed by means of a graphic user interface. If necessary, the means for changing the simulator configuration will be attached to this module.

And finally, the network connection module is responsible for providing interfaces for external applications, such as the comparative analyzer. It will also enable the simulator to transmit telemetry frames, and even receive telecommand messages, via socket services.

Implementation of this real-time simulator is an ongoing process. A prototype is being written in Python script, taking advantage of the relative ease in coding network applications using this language, and its native support to multithread programming. The idea is to take advantage of the multithreading capability of modern multicore processors to run model simulations of several different subsystems in parallel, hopefully without significant loss of performance. The porting of the planning validation simulator, originally written in FORTRAN, to Python is already complete, although tests must still be performed to check for bugs. After this task is completed, the code will be suitable for its reuse as the subsystem modules in the new simulator. Implementations of the interface modules have shown the capability to run dummy subsystems satisfactorily. Fortunately for now, there are no signs of problems that would affect the beginning of the integration process.

VI. Final Remarks

This paper described the design of a new satellite simulator by CRC-INPE, which proposal is to aid human operators in detecting anomalies during routine spacecraft operations. Because it is currently under development, no data concerning its use is available yet. However, bits of information collected during its implementation may reveal some insight about its behavior when completed, potential uses, and further developments.

First, this simulator may require a powerful, high-end machine to run properly. During the implementation of the prototype, the interface modules mockup already showed signs of significant slowdown when the simulation step was set to twice the intended telemetry transmission rate. Applications written using Python script are reputedly slower than those coded in C++, FORTRAN or Java. Moreover, for keeping large amounts of data to represent the internal state requires sufficient amount of memory. And checking trigger conditions in the events queue and interpreting system rules at every simulation step may not be possible for old or low-end machines lacking processing power. Therefore, depending on the outcome of the integrated system evaluation, porting it to a faster language may be necessary. Even if a complete overhaul can be avoided, some workarounds may be worth testing, such as the use of precompiled DLL for subsystem modules, or abandon the multithreading approach in favor of distributed computing, by running subsystem modules as independent services in different machines.

Second, the reason behind the network connection module being able to process telemetry frames and telecommand messages is its potential use with the rest of the ground TT&C software for validation purposes. Furthermore, this feature would enable the comparative analyzer functions to be incorporated into the simulator, in case its development stalls for whatever reason.

At last, there is ongoing satellite simulator development at INPE outside the CRC, using a different design approach and intended for different applications. Although unlikely to converge into the same product, future developments of one simulator may benefit from the achievements made by the other one. At the very least, lessons learned by one team could be used by the other one to avoid the same pitfalls or, to the contrary, worked even further so as to find a solution that remained unseen.

References

[1]Ambrosio, A. M., Cardoso, P. E., Bianchi Neto, J., "Brazilian Satellite Simulators: Previous Solutions Trade-Off and New Perspectives for the CBERS Program", *Proceedings of the 9th Conference on Space Operations*, AIAA, 2006.

[2]Tominaga, J., Ferreira, M. G. V., Silva, J. D. S., "A Proposal for Implementing Automation in Satellite Control Planning", *Proceedings of the SpaceOps 2008*, AIAA, 2008.

[3]Tominaga, J., Ferreira, M. G. V., Silva, J. D. S., Pereira, L. M., "Reconfigurable Satellite Simulator Modeling Approach for Extended Mission Operations", *Proceedings of the SpaceOps 2010*, AIAA, 2010.

[4]Kono, Y., Ferreira, M. G. V., Saraiva Jr., F. E. G., Pereira, L. M., "Strategies for Implementation of an Automated Planning System", *Proceedings of the SpaceOps 2010*, AIAA, 2010.

[5]Souza, P. B., Ferreira, M. G. V., Silva, J. D. S., Ambrosio, A. M., "Decision Support Tool for Prediction of Critical Data to the Satellite Integrity", *Proceedings of the SpaceOps 2010*, AIAA, 2010.

[6]Tominaga, J., Ferreira, M. G. V., "A Rule-Based Satellite Simulator for Use in Flight Operations Planning", *Journal of Computational Interdisciplinary Sciences*, PACIS, 2012.