IAC-10-B5.2.7

# INSAT3D: REAL-TIME SPACECRAFT MONITORING IN 3D

L. Claustres
VEGA Technologies SAS, France, luc.claustres@vegatechnologies.fr

E. Renaudie
Centre National d'Etudes Spatiales, France, eric.renaudie@cnes.fr

Three-dimensional visualisation provides an intuitive perception of a spacecraft's dynamics in its environment. Our aim is to demonstrate how an accurate 3D representation of the spacecraft and its internal subsystems, connected to real-time telemetry data, can also improve system monitoring for operations. Space systems are becoming more and more complex, requiring the creation of 3D models to optimise their internal design. In addition, the tools required to operate satellites are moving towards more visual displays, taking advantage of the increased computing capabilities available in control systems. We have developed a 3D monitoring tool called inSat3D, co-funded by the French space agency (CNES), which combines both trends by optimising and connecting the satellite 3D model to real-time telemetry.

In order to achieve a cost effective solution, we have chosen to base our software largely on open source components for GUI management and 3D rendering. To ensure durability, it also relies on open standards such as XML for the description of the CAD assembly or XTCE for the system database. It maps telemetry data into the model using pseudo-colour encoding to get an immediate overall, systemic and synthetic picture of the system. This is particularly useful for thermal and spatial analysis of complex systems, where traditional alphanumeric displays reach their limits. In order to help the creation of links between the CAD and the system databases, syntax matching algorithms are used along with user-driven 3D object picking. Using similar patterns, these links can be automatically or manually updated when both CAD and system databases evolve. This ensures our tool can be used from start to end of the spacecraft development process, i.e. from design to training and operations.

The ability to easily navigate in 3D through the spacecraft and all its internal subsystems provides a new efficient means to access on-the-fly contextual information for end-users. Moreover, the creation of simplified views of the system, through domain or subsystem filtering, allows them to focus on relevant status parameters and respond faster and more effectively to alarms. We believe that this tool increases the efficiency of spacecraft operations and adds value to operations training solutions. The paper will discuss these benefits, describe the development methodologies, and will summarise future plans for further development.

## I. INTRODUCTION

Ensuring the health and safety of satellites is the primary concern of operations control centres (OCC). The loss of a mission is probably the worst case, but even the more common interruption of satellite functionality can result in compromised mission objectives. For instance, the temperature inside the system should be kept within certain boundaries, not only for the payload and platform to behave properly, but also to keep them safe. As a consequence, there is a continuous improvement of integrated applications used to reduce the time required to respond to changes in a satellite's state of health. This is usually achieved through a multiple document interface (MDI) with several different kinds of display [17] including (Figure 1):

- Alphanumerics: these show parameter values as simple textual information. The output fields are colour coded to indicate the status of the parameter (e.g. red background for hard limit exceeded, yellow for soft limit exceeded).
- Line plots: these show any number of parameters against time.
- Synoptics: these show schematic functional diagrams that can include 2D graphical elements that indicate the status of associated parameters.
- Orbital displays: these show the position and attitude of the system within its environment through 2D or 3D graphical elements (Figure 2).

Although it is easy to monitor a subsystem having few parameters through these displays, engineers can hardly manage hundreds of them at the same time. The thermal subsystem is a symptomatic example of this

challenge. Indeed, there are often so many thermal sensors that engineers can hardly figure out what is the global thermal status nor the correlation between different equipment thermal behaviours just looking at the values of telemetry parameters. Moreover, the large amount of textual information makes it difficult to create a synthetic and readable display. The ability to have a graphical snapshot of the system, in order to easily inspect how the temperature is distributed within the spacecraft by 3D navigation, is a great advantage in this case. Adding sun lighting conditions to this overall picture provides the immediate comprehension and the projection of their status in the future. Thus, it can also be used for training to explain thermal related matters to engineers not familiarized with the thermal subsystem.

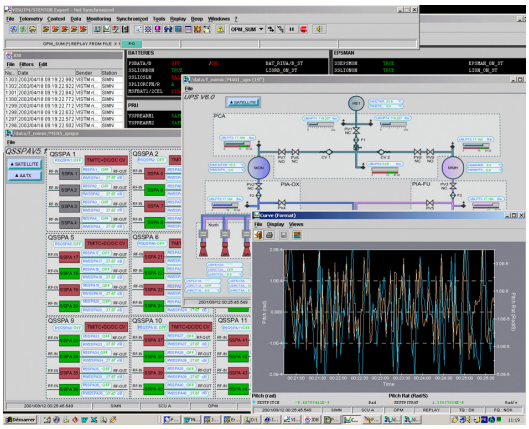

**Figure 1: MDI illustrating common displays used for operations: alphanumerics, line plots, synoptics.**

The Centre National d'Etudes Spatiales (CNES) is conducting internal research on three dimensional (3D) visualization applications to support situational awareness. The goal is to reduce the time necessary for a Spacecraft Operation Engineer (SOE) to assess the status of a satellite through an immediate overall, systemic and synthetic picture of the system. This paper describes inSat3D, a prototype software developed in the scope of these studies for the next generation of OCC displays. It also focuses on the management of information coming from disparate sources such as system databases, CAD databases and documents into an integrated three dimensional display.

The paper is organized as follows. We present the previous work that was the starting point of our research in section II. Our approach is detailed in sections III and IV by presenting the main concepts underlying inSat3D, then its architecture. The main implementation issues are discussed in details in section V. Application to the monitoring of real use cases as well as achieved performances are shown in section VI. At last, we conclude and discuss about future in section VII.

## II. RELATED WORKS

Three-dimensional visualisation provides an intuitive perception of a spacecraft's dynamics in its environment [8]. Thus, 3D display of spacecraft motion has been widely used either connected to telemetry data received from satellites in real-time [8] or ephemeris data [4]. It also improves the understanding of the state of a simulation [1], because the spacecraft is visualized as 3D object illustrating statuses like position, attitude, rotational rates, Sun direction and eclipse phases [3]. Using the ESA Mission Control System SCOS-2000 [7] the only way to visualize the results during a simulation are graphs and tables. For example the satellite's attitude, its rotational rates, etc. are visualized via plots where three curves are plotted (one for each axis). They have to be combined manually to determine the current rotational state of the satellite, while a 3D display "shows" it naturally.

At CNES, PASO (Plateau d'Architecture des Systèmes Orbitaux) is in charge of mission feasibility studies for orbital projects. This organization is composed of a dedicated engineering team and a number of specialists in various disciplines that share mission and technical constraints to achieve optimal trade-offs. In this context, a visualisation of the spacecraft's attitude during a mission's phase as a communication channel between different specialists has proven to ease the geometrical simulation [11]. It comes as an add-on for the free software Celestia©, which manages to integrate orbits and attitude data as produced by mission analysis softwares, 3D satellite models and different viewpoints for the operator, into dynamic real-time 3D displays.

Another application of 3D displays is to support collision risk monitoring activities as the growing number of orbital debris or satellites increases the probability of a collision [9]. For instance, the contingency operational procedure for collision risks management at CNES includes a dangerous conjunctions fine assessment stage involving 3D visualization [6]. The objective of this stage is to determine dangerous conjunctions that need to be mitigated and the 3D visualization helps to fully understand the geometry of the problem and validate the computation results (Figure 2).

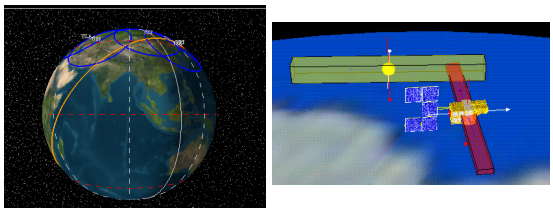

**Figure 2: Common 3D displays used for operations: orbit analysis/prediction and collision detection.**

However, our aim is to demonstrate how an accurate 3D representation of the spacecraft and its internal subsystems, connected to real-time telemetry data, can also improve system monitoring for operations. For example, the connection from the ground station to a LEO satellite can last only a few minutes, a fast conception of the current state of the satellite is thus very important in order to respond to occurring failures. As previously said, the ability to have this snapshot of the system is a great advantage for complex thermal subsystem analysis [5]. Furthermore, the visualization is helpful for the training of operators working on the project. Indeed, this 3D visualization, animated in real time, allows a much simpler approach by providing:

- an understanding of the satellite in its environment including its sub-systems;
- a synthetic and systemic vision of telemetry data;
- an ability to focus and navigate easily between subsystems in order to access information;
- a faster alarm response through the immediate link between the telemetry and the 3D model (colour code or visual effect) and the associated documentation.

## III. SOFTWARE OVERVIEW

### Graphics user interface overview

inSat3D comes as a Windows, Icons, Menus, Pointing device (WIMP) application composed of standard components such as tool bars, tool windows, menus and tool tips (Figure 3). It also includes different dockable widgets that can be freely added, removed or moved around by the user. The resulting configuration of the Graphics User Interface (GUI) is automatically saved as the user left the application. The central widget offers different *3D views* on the system and its subsystems. Each 3D view allows the user to navigate freely through the system (panning, rotating, and zooming in/out using the mouse or a compass overlay) as well as interacting efficiently and easily with the system. inSat3D uses an « observation point » as a navigation metaphor. This means that the user can turn around a fixed observation point using the mouse. This target point is moved by the user using the mouse or positioned on a specific subsystem/equipment by picking (focus).

inSat3D offers additional types of view that aim at simplifying system exploration. The *system database browser* contains a hierarchical view of the different subsystems/equipments along with a list view showing the set of Telemetry (TM) parameters of the currently selected subsystem/equipment. Filtering can be applied in order to limit the potentially large listing to parameters matching a given syntax scheme, unit (degrees, radians, etc.) or type (enumeration, float, etc.). These widgets are used to manage the system from an operational point of view. Most user actions are possible from these views through a context menu on the elements of the system tree or the parameter list. The *CAD database browser* includes a hierarchical view to explore the CAD assembly (i.e. the tree of 3D objects), which is mainly useful when preparing data (see §IV). Usually, this view is not used in operations because users interact through the system or the 3D views. The CAD view shows the different grouping nodes of the assembly, as well as the associated parts (3D objects). A subset of the user actions is available from this view through a context menu on the elements of the assembly tree. However, most actions of this menu are also possible by picking an object on the 3D view in order to simplify usage. From the CAD browser, the 3D view or the system database, links between selected elements can be indifferently created, edited or deleted. A menu bar composed of different tool bars, including drop-down lists, completes the GUI with actions corresponding to the main application functions:

- working environment creation, loading, closing or saving;
- system and CAD database import and update;
- telemetry data sources selection;
- visualisation mode switching (from standard lighting to sun shadowing).

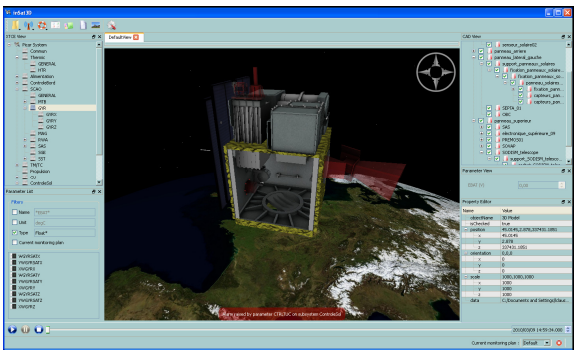

**Figure 3: The GUI of the inSat3D application, the system database browser is on the left, the 3D**

**view in the centre and the CAD database browser on the right.**

Features overview

As a monitoring tool, inSat3D allows displaying the values and the status (validity states, alarm states) of telemetry parameters to the user. However, it uses the specificity of a 3D view in order to provide new ways of interaction. Indeed, inSat3D allows the observation of the satellite, including its subsystems/equipments, from all angles and can manage pre-defined viewpoints. The visibility state (hidden, shown or transparent) of subsystems or equipments can be changed on-the-fly, and this configuration stored per view. This leads to the creation of simplified views of the system, through domain or subsystem filtering, allowing the user to focus on relevant status parameters and respond faster and more effectively to alarms. The underlying 3D viewer does support multi-views and multi-windows in order to manage these features (Figure 4).

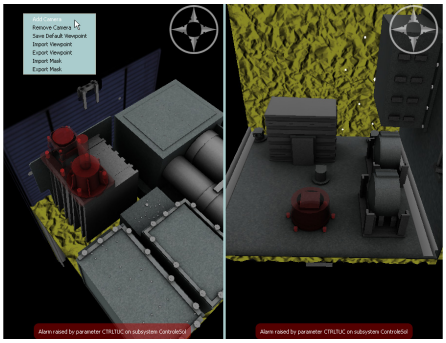

**Figure 4: Multi-views capability of inSat3D.**

The main objective of inSat3D is to relate or *link* the visual attributes of 3D objects to parameter values. These attributes can be the following: the colour, the opacity/transparency and the visibility. Animations can be done by relating telemetry parameters to object's orientation (rotation angles or attitude quaternion). The user can then define the reference frame in which the rotation has to be performed (origin and axes). Indeed, the mechanical (or CAD) frame can be different from the one used to express the parameter in the system database. The temperatures values can be shown both in a textual and in a visual way (Figure 5). The visual information consists of colour-coded spheres (or cubes) in the spacecraft that represent the location and temperature of the thermal sensors, if not available in the CAD model (Figure 5). Otherwise, the telemetry parameters can be directly linked to the 3D representation of the equipment or the sensor (Figure 6). The different shapes of the virtual sensors are used to visually categorize thermal sensors and heaters.

inSat3D comes with a friendly user interface to create the links between the different databases (Figure 7): 3D database, system database and document database. For parameters the GUI allows to select the colour/opacity gradient that will be related to the parameter variation range (e.g. [-5, 5]). Similarly, the GUI allows selecting the colours related to the different parameter values for enumerations (e.g. ON/OFF). The localization of the sensor providing the parameter can be chosen by 3D picking onto the model (in case the sensor is not a self-defined 3D object, it will then be represented as a small sphere). For documents, a file chooser is provided to select the target file associated with an element. The links between subsystems and 3D objects are done through drag & drop operations between the system view and the 3D view or by object picking on the 3D model.

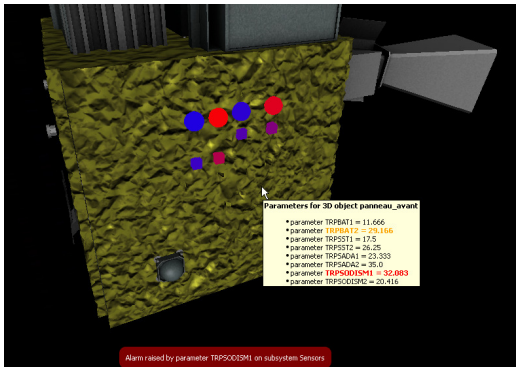


**Figure 5: Virtual thermal sensors attached to a satellite's panel. The associated tooltip showing instantaneous temperature values appears when the user move the mouse over the 3D object.**

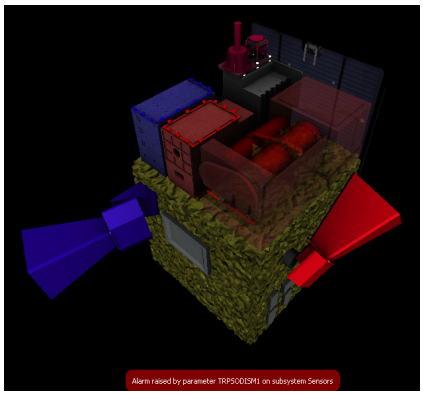


**Figure 6: A snapshot of the thermal state of a set of equipments on a satellite. The transparent one is blinking because it comes close to its safety boundary (alarm has been raised by monitoring).**

inSat3D allows to display the parameter values related to a subsystem/equipment as a 2D text overlay or in a dedicated window where a selected set of

parameters can be dropped on the fly. Depending on the current alarm status of the parameter, its text colour will be chosen accordingly (green for nominal range, yellow for the caution level and red for the action level). Similarly, attached 3D objects will blink accordingly until the alarm has been closed on the related parameter. inSat3D also provides access to corresponding context documentation (URL or PDF file defined by the user) by 3D object picking, parameter/subsystem selection in the system view or a hyperlink overlay that automatically appears when an alarm is raised (Figure 8).

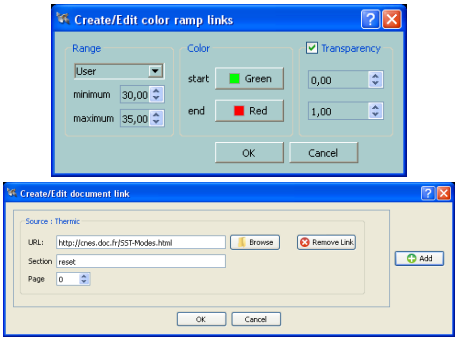


**Figure 7: Example of available graphical link editors in inSat3D.**

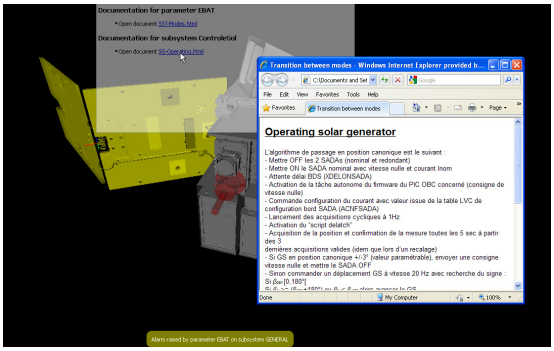


**Figure 8: Example of context documentation automatically shown by inSat3D when an alarm is raised. The user can activate hyperlinks to open the target document files (HTML, PDF, etc.).**

Because users rely significantly on the visualization to interpret the system state, it is essential that the visualization model correspond accurately to the real geometry. For this reason, inSat3D can build its internal 3D model directly from an existing CAD model. However, keeping graphics models consistent with design models is a challenging task as the system structure and design evolves especially so during the early phases of projects. To mitigate this, inSat3D uses an identifier referencing system that is faithful to the underlying elements. 3D graphics parts of the satellite (such as wheels, masts, arms, etc.) or documents are not directly attached to the subsystems/parameters of the system database but through external link objects. This weak coupling between databases ensures that the design and visualization models can maintain close correspondence to each other. Indeed, when either the CAD database or the system database evolves, source (subsystems/parameters) and target elements (3D objects) of a link are automatically updated based on the reference name. Moreover, by managing links as tangible objects, the user is able to view current link status and manually update it if pointing to invalid source and/or target elements (Figure 9). This can particularly occurs when an element is deleted or renamed from the input database.

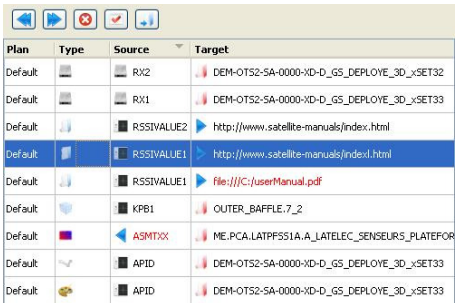


**Figure 9: Through the link browser the user can visualize invalid links (in red), multi-select existing links and categorize them according to type, source and target elements in order to simplify manual updates.**

Provided that the data source supports playback of data, inSat3D can also control playback time and speed via a set of buttons similar to a tape or video recorder control panel. The buttons can be used for pausing playback, changing playback speed or resuming real-time data display. In addition, it is possible to enter a playback time manually or to move a slider on a time line. This is a very useful feature allowing offline analysis of anomalies on some consoles whilst real-time operations continue elsewhere.

## IV. SOFTWARE ARCHITECTURE

The general inSat3D architecture is based on the separation between the data setup phase and the operations phase. The data setup phase consists in importing/updating input data (the CAD model and the system database) and creating/checking the links between the different databases in order to build a working environment. The operations phase consists in connecting this environment to telemetry providers in order to monitor the system. Once done, the render engine is in charge of updating the visual attributes of the system whenever a parameter status evolves. The value of a visual attribute is computed according to the

user-defined behaviour (i.e. the set of active links) based on the current parameter value provided by the telemetry data source. The application reacts to user interaction through the 3D view (navigation, picking) as well as the 2D GUI (selection, edition). The general architecture diagram is shown Figure 10.

These tasks were given different user's privileges or role even though it can be actually performed by the same person. According to the user's privileges, specific actions are enabled or disabled in the inSat3D application. At the present time three roles have been identified:

- the *operator* can open existing environments and select telemetry sources to perform system monitoring;
- the *expert* can additionally edit existing working environments;
- the *administrator* can additionally create (or update) working environments based on input system/CAD databases.

External and internal data file formats

inSat3D can retrieve data coming from existing CNES softwares through different file formats. For instance, the Catia CAD software is mainly used to design satellites. Thus, the tool can convert a Catia V5 assembly to the native file format of the underlying 3D engine Open Scene Graph (OSG) for efficient visualisation. Concerning the System Database (SDB), the exchange format used is the XML Telemetry and Command Exchange (XTCE) [12]. Indeed, it allows retrieving the hierarchical decomposition of the system in terms of subsystems/equipments, as well as the description of the attached set of TM parameters (including ground alarms). As CNES is on the way to integrate even more space community standards in its product line with the ultimate goal of spacecraft operations systems unification, XTCE seems to be the best choice for softwares importing SDB in future [16].

Application-specific data are the links created between the different databases. These links are grouped together into different *monitoring plans*, which can be activated on user's request. Thus, each subsystem or equipment can be decorated with a coherent set of links toward other databases (CAD assembly and documentation). The eXtensible Markup Language (XML) [14] is internally used by the application to store user environments, i.e. the SDB with the associated monitoring views and plans created by the user. This format has the advantage to be easily read by both computers and human beings. This format also makes it easy for configuration utilities to be written which can generate inSat3D environment definitions from environment defined in other formats or through an automated process. Figure 11 summarizes the different data formats managed by the application.

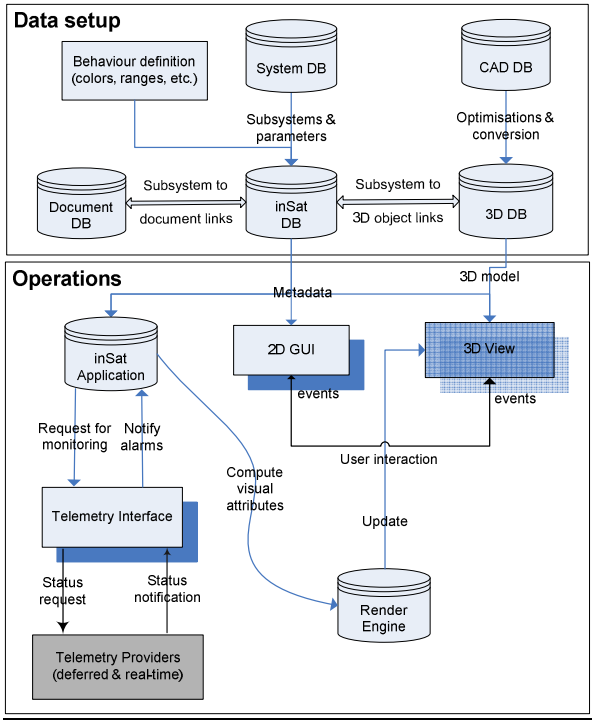


**Figure 10: inSat3D general architecture diagram.**

inSat3D is also able to read decommuted TM data and associated monitoring data from archived text files in order to simulate real-time data flow (i.e. deferred time) through its playback (play/pause/stop/jump) interface. This data flow can also include celestial body positioning information (earth, sun and moon) from ephemerid files.

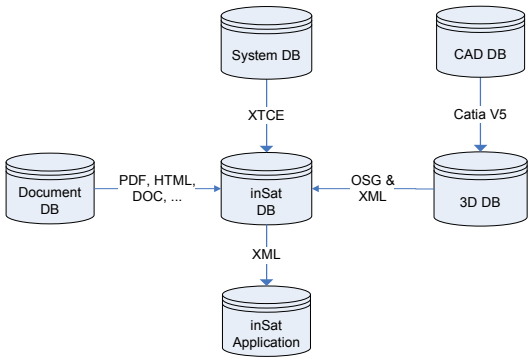


**Figure 11: The different file formats used by inSat3D for external and internal data management.**

## V. DETAILED SOFTWARE PRESENTATION

This section focuses first on some of the main inSat3D features: the CAD extraction process, the

system database management and the telemetry provider interfaces. The underlying technologies used to build inSat3D will be detailed in a second phase.

CAD data extraction

The main objective of the CAD extraction (Figure 12) is the processing of a digital satellite model coming from design in order to produce an optimized 3D model well-suited for real-time rendering. The main idea is to retrieve and merge the different CAD surfaces related to a specific part (e.g. stellar sensor) into a single output 3D mesh (i.e. object) for optimal performances. The accuracy for the approximation of input surfaces into polygons is defined in terms of model units (e.g. 1mm). This is used to control polygonal mesh density for an optimal visual quality/rendering performance trade-off. Specific optimizations to increase real-time performance can also be done such as removing hidden objects, removing small objects based on size, joining equal points, etc. Specific filtering is also useful to remove unnecessary parts from the CAD model for operations (such as screws) and ensure optimal performances. Assembly structure can also be simplified.

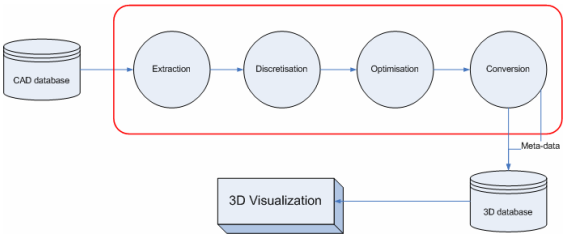


**Figure 12: The CAD extraction process.**

Once the 3D model has been created, it is converted to the native input format (OSG) of inSat3D with additional information such as assembly node/part names, default colours, etc. Indeed, the CAD extraction process takes place once and further operations are directly performed on the processed model. Even if this automated process can generate a plausible and real-time suited 3D model, it remains constrained by the input CAD model quality/complexity. Thus, optimizations and filtering can also be manually applied through additional and dedicated 3D modelling softwares. It is also possible to integrate default colours or textures in the 3D model. However, CNES recommends not to use such default visual attributes to avoid interpretation. Indeed, it can be complicated for an SOE to know if the object appears blue because of its default colour or because of its current state (temperature, voltage, etc.).

System database extraction

The main objective of the system database extraction is the processing of the system database in order to produce a parameter database well-suited for inSat3D. The COntrol Centre Parameters Ingest & Treatment (COCPIT) software used to manage SDB at CNES is able to export a SDB in the XTCE file format. However, at the present time, it still requires a transformation in order to match the internal data model based on the following hierarchy: functional domain → subsystems → equipments → parameters. Indeed, the exported database is a flat list of parameters attached to a single space system, with additional information about the associated subsystem but nothing concerning the associated equipment. Moreover, the system decomposition is usually specific to a particular mission and does not fulfil a particular standard such as ECSS-70-31 [13].

Although the crude database exported from COCPIT can be used as valid input XTCE for inSat3D it is not very convenient. On the one hand, the database need to be converted from a flat version, where all parameters will be listed in an unordered manner, to a hierarchical version based on the subsystem information, which makes the system exploration easier for the user. On the other hand, the mission-specific decomposition has to be converted to a more generic system decomposition. As this transformation is a data-centred process, we have chosen to implement it using Extensible Stylesheet Language Transformations (XSLT), which is a declarative, XML-based language used for the transformation of an input XML file to another XML document [15]. Our XSLT stylesheet contains the target hierarchical structure of the system, built using the recursive XTCE *SpaceSystem* element. This element is interpreted as a grouping node for child parameters by inSat3D, whatever the hierarchical level (subsystem or equipment), thus providing efficient filtering based on the user selected level. Template rules are defined to select and move parameter nodes matching a given pattern in the input file into a particular target space system. These rules allow to perform simple name correspondence between both the mission and the generic decomposition, but also to create the missing equipment level. For instance, all parameter names starting with “RW” followed by a number N (such as “RW1_TEMPERATURE”) will become parameters of the wheel number N.

Telemetry data providers

The architecture of inSat3D is based on a strict separation between the data providers (data sources) and the user interface. This separation ensures a great flexibility and enables to present data from many different back-end data providers. Each provider is

required to implement a standard interface which is then used by the application to:

1. query the source for the parameters it provides;
2. register an interest in receiving updates for a set of parameters;
3. query the source for the status of a specific parameter.

inSat3D uses a service-oriented consumer-provider API, inspired from the SM&C CCSDS standard [10], which simplifies the integration of additional data providers. Indeed, the 3D visualization has simply to be registered as a consumer of the new data provider to connect the 3D model to the telemetry data flow. Moreover, the migration to future generic control centre architectures will be easier. The receiving process can be performed in a pushed way (**push** mode), i.e. the consumer is activated asynchronously by the provider whenever a parameter status changes. The receiving process can also be performed in a pulled way (**pull** mode), i.e. the consumer requests synchronously the provider for parameter status. The C++ provider API is summarized hereafter:

- `registerForMonitorStatus(consumer, parameter list)`: register a consumer to the provider for the given list of parameters (only status updates for this parameter set will be notified to the consumer);
- `deregisterForMonitorStatus(consumer, parameter list)`: deregister a consumer from the provider for the given list of parameters (status updates for this parameter set will not be notified anymore to the consumer);
- `requestDefinition(parameter)`: retrieve the information associated to a parameter such as description, monitoring interval, type, unit;
- `requestStatus(parameter)`: retrieve the current status (value, alarm checking) of the given parameter.

The C++ consumer API is summarized hereafter:

- `notifyMonitorStatus(provider, parameter status)`: notification sent by the given provider when the given parameter status has changed.

At the moment, the single data provider of inSat3D is implemented in term of simulated real-time data flow read from telemetry archive files (play/pause/stop control commands). These files describe the incoming parameter values for each telemetry packet received within a frame. These files also include monitoring information as timed alarm events (list of low/high caution status at given date/time).

Scripting engine for test automation

Although GUIs make software easy to use from a user's perspective, validation of software such as inSat3D is more complex than testing conventional software. The difficulty in accomplishing this task is twofold: domain size and sequences. In addition, the tester faces more difficulty when they have to do regression testing. Unlike a command line interface (CLI) system, a GUI has many operations that need to be tested (a simple program such as WordPad has 325 possible operations [18]). Moreover, functionalities of the software are accomplished by following a complex sequence of GUI events. This can become a serious issue when the tester is performing test cases manually. At last, because the GUI may change significantly across versions of the application, regression testing also becomes a problem. A test designed to follow a certain path through the GUI may not be able to follow that path since a button or a menu may have changed its location.

These issues have driven the GUI testing problem domain towards automation. Many different techniques have been proposed from capture/replay tools to test programming frameworks. The way to run tests on inSat3D makes use of a built-in driver into the GUI so that commands or events can be sent to the software from another program [19]. This method of directly sending events to a system is highly desirable when testing, since the input and output can be fully automated and user error is eliminated. It has been accomplished through the use of a scripting engine based on the ECMAScript scripting language, as defined in standard ECMA-262 [20]. Specific script commands have been defined for each possible user actions: loading telemetry data, creating links, etc. Moreover, any inSat3D object instance (views, monitoring plans, links, etc.) can be made available for use within scripts. The properties of the objects are available as properties of the corresponding script object. When you manipulate a property in script code, the C++ get/set method for that property will automatically be invoked. At last, a screenshot of the 3D view can be taken on-demand in order to get a snapshot of the current system state according to monitoring. This image can be automatically checked against a previously stored reference image for further regression testing.

Underlying technologies

VEGA proposed to answer to CNES requirements by using a combination of open source components for GUI management and 3D rendering, bringing out the

best of each into the solution. The choice of these components was based on extensive research to understand their strengths, weaknesses, popularity, and community support. Moreover, these components have been intensively used across industrial projects, assuring scalability and stability as well as continuous improvement. On top of this robust layer, VEGA enhances required functionalities and builds user-friendly tools. Advanced GUI providing simple operations (such as drag & drop), are used whenever possible. This approach has been preferred by CNES to Components Off The Shelf (COTS) in order to achieve a recurrent cost effective solution and a tailored software that exactly matches the requirements.

3D rendering

This section briefly introduces the main component of inSat3D: the 3D viewer. This viewer is based on Open Scene Graph (OSG) (www.openscenegraph.org), an open source high performance 3D toolkit allowing a level of quality/realism comparable to what is available on the market with top level serious games. OSG runs on all Windows platforms, OSX, GNU/Linux, IRIX, Solaris, HP-Ux, AIX and FreeBSD operating systems. OSG is now well established as the world leading scene graph technology, used widely in the vis-sim, space, scientific, oil-gas, games and virtual reality industries. Well-known French users are the following: « Société Nationale des Chemins de Fer » (SNCF), « Bureau des Ressources Géologiques et Minières » (BRGM), « Centre National d'Etudes Spatiales » (CNES), « Office National d'Etudes et de Recherche Aéronautique » (ONERA) and « Direction Générale des Armées » (DGA).

The 3D viewer also integrates the Visualization Toolkit (VTK) (www.vtk.org), an open source library used by VEGA for post-processing of multi-dimensional meshes. It can be used to perform data extraction (cutting planes, iso-lines, iso-surfaces, etc.) or data interpolation (colour mapping onto a 3D mesh for instance). The viewer combines both libraries through high-level functionalities. Indeed, processing is performed by VTK while the rendering of the extraction/extrapolation results are viewed through OSG. Although this toolkit might be useful to display temperature gradient on a satellite's surface, it has not been used in the frame of the first mission targeting to use inSat3D because of the lack of thermal sensors. However, this capability could deserve more complex spacecraft such as the Automated Transfer Vehicle (ATV) featuring hundreds of sensors.

Graphics User Interface

Qt is a cross-platform application and UI framework supported by Nokia (http://qt.nokia.com) and used by VEGA to develop the inSat3D GUI. As a market leader, Qt has been widely used for over 15 years to build innovative software applications that run across multiple desktop operating systems and embedded devices (all Windows platforms, OSX, GNU/Linux, IRIX, Solaris, HP-Ux, AIX, etc.). More than 5,000 companies from a broad spectrum of industries – including Google, Adobe, Lucasfilm and Skype – use it. Qt provides integrated tools to boost application development such as a rich library of standard widgets, a dynamic layout engine, a network communication layer (client and server socket abstraction for common protocols), XML standard management (XPath, XQuery, etc.), multi-core architecture programming, translation and internationalization tools.

System database extraction

Because pre-processing of XTCE databases, as detailed previously, is not a mandatory task for application usage, we have chosen to rely on external XSLT processors. Indeed, a lot of open source processors are now available as standalone products, or as components for others softwares including web browsers, application servers, or even operating systems.

CAD data extraction

This task is performed through the use of the CAD Import Manager software developed and provided by Global Vision System. Indeed, CNES requirements included the ability to directly manage the native format of its CAD data, which is currently Catia V5, in order to avoid additional work for the designers. However, this proprietary and closed file format can only be read through proprietary softwares. Future versions of inSat3D will also include a reader based on open source components for CAD exchange file formats such as STEP or IGES.

## VI. USE CASES AND ACHIEVED RESULTS

The PICARD mission has been selected by CNES for the development of inSat3D. This mission is an investigation dedicated to the simultaneous measurement of the absolute total and spectral solar irradiance, the diameter and solar shape, and to the Sun's interior probing by the helioseismology method. These measurements obtained all along the mission will allow to study their variations as a function of the solar activity. Its objectives are to improve our knowledge of the functioning of our star through new observations and the influence of the solar activity on the climate of the Earth.

Performances

inSat3D is able to display 3D models directly coming from design thanks to its high performance 3D engine managing both *frustum* and *contribution* culling. The goal of frustum culling is therefore to be able to identify what is inside the frustum* (totally or partially), and cull everything that is not. Only the 3D objects that are inside the frustum are sent to the graphics hardware. Thus, the graphics hardware solely renders what is potentially visible, saving on the processing of all those objects that are not visible anyway. Furthermore, this can potentially improve the performance of the application since only the objects that are part of the visible part of the 3D model are kept on the graphics card memory, and these are more likely to fit than the whole 3D model. The goal of contribution culling is to discard objects if their screen projection is too small (in practice, the projection of their bounding volume and a user-defined threshold is used). This form of culling isn't conservative, but is still very interesting because CAD models often include many small parts such as screws, wires, etc. that do not contribute significantly to the final image. At last, culling and drawing can be performed on a separated application thread (these are read-only operations) on multi-core architectures to increase performances.

Even accurate 3D models generate an acceptable amount of data after conversion (Table 1). However, for best performances and accuracy it requires a high-end 3D graphics processor on the client machine. Indeed, tests demonstrated that even an accurate (0.05mm tolerance) and not optimized 3D model of PICARD runs well with a dedicated 3D chipset but requires dramatic simplification to perform similarly without any 3D chipset. This is shown by the Frames Per Second (FPS) performances illustrated in Table 2. Results presented hereafter were achieved using an AMD Dual Core CPU at 2GHz and an ATI Radeon HD 2600 Pro graphics processor.

| Accuracy (mm) | Optimi-sations | Size (MB) | # of vertices | # of polygons | # of nodes |
|---|---|---|---|---|---|
| 0.05 | none | 85.5 | 3 M | 1 M | 2124 |
| 0.05 | merge | 43.5 | 824 K | 1 M | 2124 |
| 0.5 | none | 30.6 | 1 M | 350 K | 2079 |
| 0.5 | merge | 18.1 | 350 K | 350 K | 2079 |
| 0.5 | merge + filter | 17.5 | 350 K | 350 K | 1997 |
| 1 | merge | 14.2 | 285 K | 270 K | 1634 |
| 10 | merge + filter | 1.1 | 17 K | 10 K | 347 |

**Table 1: CAD data conversion output size for different accuracies and optimisations.**

* The view frustum is the volume that contains everything that is potentially visible on the screen, defined according to the current camera's (i.e. observer's) settings.

| Accuracy (mm) | FPS (with 3D chipset) | FPS (without 3D chipset) | Loading time (ms) | Processing Time (m) |
|---|---|---|---|---|
| 0.05 | > 60 | 1.5 | 828 | 6 |
| 0.05 | > 60 | 2.2 | 625 | 10 |
| 0.5 | > 60 | 3.2 | 625 | 6 |
| 0.5 | > 60 | 4.5 | 406 | 8 |
| 0.5 | > 60 | 4.5 | 391 | 9 |
| 1 | > 60 | 5.3 | 328 | 7 |
| 10 | > 60 | 24 | 156 | 6 |

**Table 2: Achieved performances on the 3D model depending on the CAD data accuracy.**

Use cases

inSat3D has been developed and tested according to different use cases, from thermal state analysis to geometry related issues such as panel deployment or sun facing manoeuvres. But PICARD has been successfully launched and brought in orbit at the present day. The first telemetry from the satellite arrived as expected, and is currently used to validate the software with real mission use cases too. As an example, Figure 13 illustrates operations that bring the payload to operational configuration. Particularly, the SODISM's CCD is being heated to increase outgassing in order to decontaminate it. At the end of this stage it should reach the mean temperature of 25°C. This use case demonstrates the simplified view capability of inSat3D by focusing on the payload only, filtering everything else. Each equipment is linked to its temperature but the SODISM (shown on the right side of the images) also includes three virtual sensor connected to acquisition points on its support. The visual behaviour has been configured to provide a global colour gradient on the payload between dark blue (cold) and red (hot) for the temperature range [-25°C, +25°C].

Figure 14 illustrates operations that bring the satellite from geocentric to heliocentric pointing. Depending on the position of the earth, moving around during this transition, the on-board software selects which stellar sensor to be used at a given time. Moreover, the temperature of the involved equipments as well as side panels slightly varies. In this example, inSat3D is configured to simultaneously view the temperature of both stellar sensors and panels using its multi-viewpoint capability. Using filtering, another panel is also removed in order to view internal equipments and ease battery voltage monitoring. At last, depending on the currently selected stellar sensor the other one is made transparent.

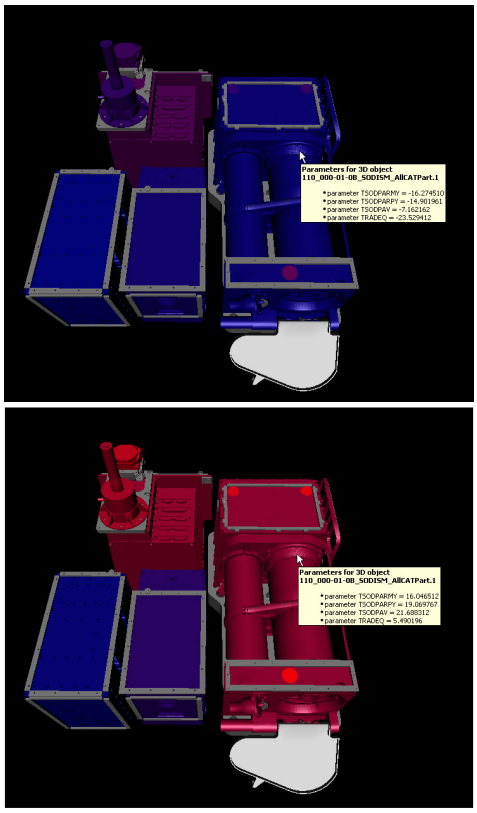


**Figure 13: A 3D view of the payload's thermal state before (top) then after (bottom) the decontamination of PICARD's SODISM. The user can display instantaneous temperature values by picking the target equipment.**

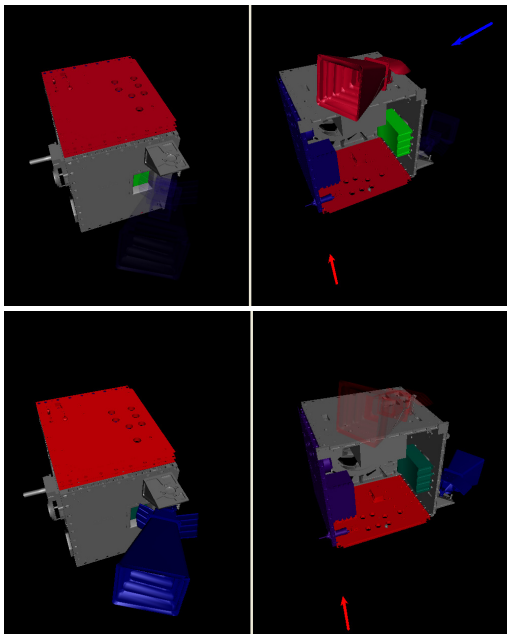

**Figure 14: Different configurations of the geo- to helio-centric pointing transition. The first (top) then the second (bottom) stellar sensor is alternatively used depending on the earth position. The red arrow is the sun direction while the blue one is the earth direction.**

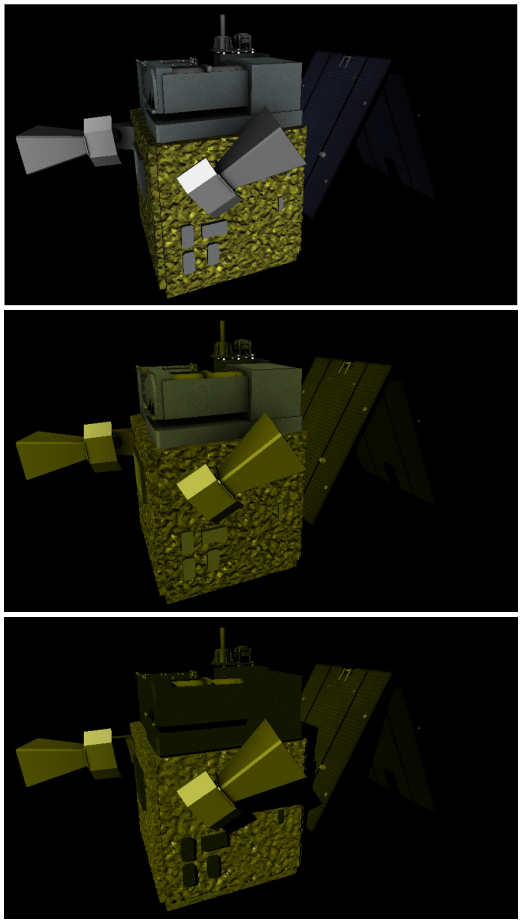

**Figure 15: A 3D view comparison including standard lighting (top), sun lighting (centre) then self shadowing (bottom). The sun light is coming from the top-left of the image.**

The implementation of an advanced lighting model for enhanced scene realism, including sun lighting as well as self shadowing, has proven to be a useful analysis tool for complex thermal problems. Indeed, it can be used to detect cold spots due to the shadow of an equipment on another. This is for instance illustrated in Figure 15 (respectively Figure 16), which shows the satellite during (respectively after) the solar array deployment sequence. This lighting model benefits from the impressive improvements in the capabilities of graphics processing units (GPUs) for the past few years. Indeed, among many features, the most fascinating achievement is the realisation of GPU programmability for real-time high quality local illumination by directly evaluating physically-based reflection models as each

pixel is shaded. Our vertex and fragment shaders are written in the GLSL shading language managed through the OpenGL 2.0 API by OSG.

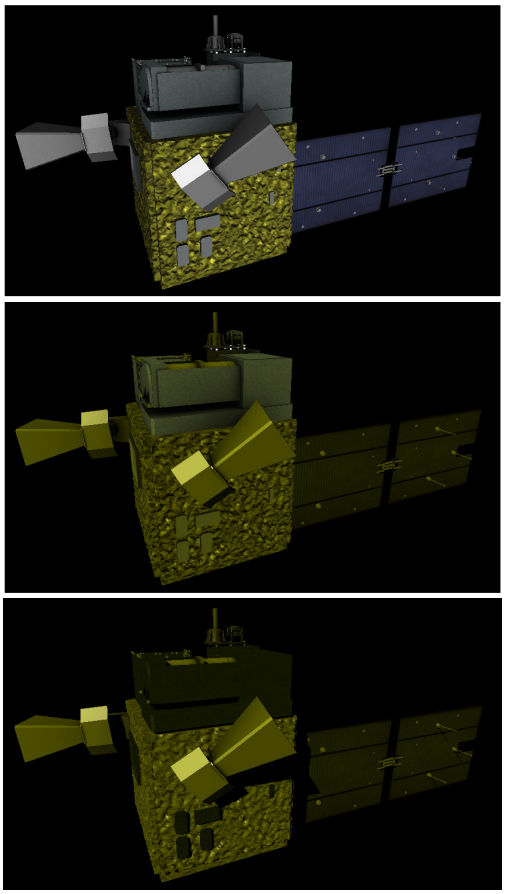

**Figure 16: A 3D view comparison including standard lighting (top), sun lighting (centre) then self shadowing (bottom). The sun light is coming from the top-left of the image.**

## VII. CONCLUSION AND FUTURE WORK

By mapping telemetry data into a realistic 3D model using pseudo-colours, inSat3D provides an immediate overall, systemic and synthetic picture of the system. The ability to navigate through the spacecraft and all its internal subsystems simplifies access to contextual information for end-users. Moreover, simplified views of the system, through domain or subsystem filtering, allows them to focus on relevant status parameters and respond faster and more effectively to alarms. Working environments can be automatically or manually updated when both CAD and system databases evolve. This ensures inSat3D can be used from start to end of the spacecraft development process, i.e. from design to training and operations. We believe that it increases the efficiency of spacecraft operations and adds value to operations training solutions. At last, inSat3D is a durable cost effective solution, largely based on open source components and open standards.

inSat3D development is ongoing and future plans include the implementation of a client-server interface that will allow users to directly connect the software to available data providers at runtime. inSat3D will also provide orbital and time-related displays, including earth/moon/star modelling, for enhanced scene realism. More specifically, for CNES use, it will provide interconnection with existing orbital 2D/3D displays. However, the main challenge that inSat3D want to tackle is determining the satellites state of health as fast as possible. Talking with operators and observing operational procedures has shown that operators needed access to different display types (line plots, alphanumerics, synoptics, etc.) once a deviation from expected satellite behaviour was detected. An unnecessary delay in anomaly resolution is incurred as operators are forced to move between multiple terminals and documents to gather, sort, and merge the independent sources of information. By simply automating the collection of existing information and creating a single point of access for operators, a significant improvement to satellite monitoring and protecting could be realized. We believe that the 3D representation is, by its intuitive and synthetic nature, a perfect entry point. By further providing a three dimensional display including interactive access to other kind of displays (1D, 2D) in addition to standard documentation, a system could efficiently cue the human brain to highly correlated and time critical information. We will particularly investigate how we can make use of the implicit spatial map of the system database on the 3D model to generate interactive synoptics on-the-fly through 3D screenshots or 2D projections connected to alphanumerics. Through simplified views of the system, created using domain or subsystem filtering already available in the software, even synthetic synoptics with parameters coming from unrelated locations could be generated.

## REFERENCES

[1] “Dspace: Real-time 3D Visualization System for Spacecraft Dynamics Simulation”, Marc I. Pomerantz and Abhinandan Jain, Third IEEE International Conference on Space Mission Challenges for Information Technology, pp.237-245, 2009.

[2] “3D Visualization of Satellite Status and Environment Using Open Source Software”, Leeha R. Herrera, Nigel H. Tzeng and Osbaldo Cantu, SpaceOps 2006 Conference.

[3] “Real-Time 3D-Visualization in Satellite Development”, Witt R., Fritz M., Kuwahara T., Brandt A., Laurel C., Röser H-P, Eickhoff J., 4th International Conference on Astrodynamics Tools and Techniques (ICATT), 2010.

[4] “3D Graphical Simulations for Spacecraft Missions”, Crosnier T., 4th International Conference on Astrodynamics Tools and Techniques (ICATT), 2010.

[5] “Virtual Reality for Monitoring Spacecraft Thermal Subsystem.Concept & Prototype”, A. Donati, J.A. Martínez-Heras, P. Nunes and R. Torrão, SpaceOps 2004.

[6] ”Operational management of collision risks for LEO satellites at CNES”, F. Laporte and E. Sasot, ?.

[7] “Simulating Spacecraft Systems”, J. Eickhoff, Springer Aerospace Technologies Series Vol. 1, Springer Verlag, Berlin-Heidelberg, Germany, ISBN: 978-3-642-01275-4.

[8] ”3D Display Of Spacecraft Dynamics Using Real Telemetry”, L. Sanguk, C. Sungki, and K. Jaehoon, J. Astron. Space Sci. 19(4), pp. 403-408, 2002.

[9] ”On the road to operational monitoring of collision risks in space”, F. Laporte, SpaceOps 2000.

[10] ”Spacecraft Monitoring and Control – Core Services, Red book”, CCSDS 522.0-R-2, 2008.

[11] “Axes of progress engaged at CNES to improve capacity of collaboration with partners between Concurrent Design Facilities”, Le Gal JL., Warrot T., Joubert M. and Haardt E., 10th NASA-ESA Workshop on Product Data Exchange, 2008.

[12] ”XML Telemetric and Command Exchange (XTCE)”, CCSDS 660.0-B-1, 2007.

[13] ”Ground systems and operations - Monitoring and control data definition”, ECSS-E-ST-70-31C, 2008.

[14] ”Extensible Markup Language (XML) 1.0”, Fifth Edition, W3C Recommendation, 2008.

[15] ”XSL Transformations (XSLT) Version 2”, W3C Recommendation, 2007.

[16] ”CNES Spacecraft Operations System Roadmap”, M.L. Anadon, N. Champsavoir, P. Gelie, E. Poupart and H. Pasquier, DAta Systems In Aerospace (DASIA), 2010.

[17] “SATMON – A Generic User Interface for Satellite Control”, C. Peat, SpaceOps 2004.

[18] “Using a Goal-driven Approach to Generate Test Cases for GUIs”, Atif M. Memon, M.E. Pollack and M.L. Soffa, Proceedings of the 21st international conference on Software engineering, pp. 257-266, 1999.

[19] “Toward automatic generation of novice user test scripts”, D.J. Kasik and H.G. George., Proceedings of the Conference on Human Factors in Computing Systems : Common Ground, pp. 244-251, 1996.

[20] “ECMAScript Language Specification”, Standard ECMA-262 5th edition, 2009.