IMIS
Solutions for assessing instruments' health from anywhere

Olivier Queyrut[1], Julien Baroukh.[2], Julien Airaud[3], Grégory Pradels[4]
Centre National d'Etudes Spatiales, 31401 Toulouse, France

Since the advent of mobile technologies (laptops, smartphones and tablets) as well as the worldwide deployment of broadband networks (3G, Wi-Fi hotspots), the demand for accessing the satellite operations centers from outside has increased. Until now, very few means to carry out this demand have been set-up at CNES. As part of the MSL mission (Mars Science Laboratory), one of the strong requirements of the CNES operations center for the ChemCam and SAM instruments (the French contributions to MSL instruments) was to enable users outside the operations center, but of course with sufficient privileges, to assess the instruments' health. CNES then implemented new software solutions named IMIS (Instrument Monitoring Interactive Software) which provide all features necessary to monitor the payload instruments from anywhere. In addition to a desktop client application, IMIS also provides an Android application mainly designed for on-call operations.

I. Introduction

In the CNES payload operations and data centers, few operational means are currently available from outside the center. In most cases, only access to the catalog of products and downloads are possible outside CNES via a Web interface. The laboratories that develop a payload instrument have no way to remotely monitor the technological parameters of their instrument. Three reasons can explain this fact. Firstly, the distributed architecture and the network technologies are actually quite recent (early 1990s) on the scale of a satellite project. Then, the needs for remote access rise with the modification of working habits and with the wide use of the Internet in everyday life that did not exist a few years ago. Finally, the justified concerns for the information system security were an impediment to the development of distributed applications over the Internet. Indeed, confidentiality, data integrity, and availability which are key factors of any operational center, can be threatened by external accesses from uncontrolled environment.

As part of the MSL mission, one of the strong requirements of the CNES operations center for the ChemCam and SAM instruments was to enable users outside the operations center, but of course with sufficient privileges, to assess the instruments health. Fortunately, thanks to political and technological changes in matter of security, some applications can now be available on the Internet provided that the sensitivity of the data that are transmitted and handled, allows their dissemination. This is specifically relevant for a scientific mission which is operated in collaboration with laboratories and thus requires openness to the Internet.

New software called IMIS was thus developed to meet this requirement. IMIS is firstly used in the frame of the MSL and SSALTO projects but it can also be integrated in any CNES scientific payload data and control center. Moreover, due to the great success of mobile technologies (Smartphones and tablets) and the extended capabilities of broadband networks (3G and Wi-Fi hotspots), the idea of developing an IMIS client for Android platforms has emerged. This application named SmartMonitorer is dedicated to on-call operations allowing to remotely perform a first level analysis when an alarm is raised (because a telemetry parameter is for example out of range).

After the presentation of the IMIS operational requirements declined for MSL and SSALTO missions and although for on-call operations (section 2), the IMIS functions and design are detailed in sections 3 and 4.

[1] Ground Segment Engineer, CNES/DCT/PS/CMI, olivier.queyrut@cnes.fr.
[2] Ground Segment Engineer, CNES/DCT/PS/CMI, julien.baroukh@cnes.fr.
[3] Security Engineer, CNES/DCT/ET//SSI, julien.airaud@cnes.fr.
[4] Ground Segment Engineer, CNES/DCT/PS/CMI, gregory.pradels@cnes.fr.

II. Operational requirements

A. Generic Payload Control Center

The ground segment typically consists of the following top‑level systems which can be distributed across various centers depending on the chosen ground segment architecture [ECSS-E-ST-70]:

- Mission operations system;
- Payload operations and data system;
- Ground station system;
- Ground communications system.

The payload operations and data system (also called 'payload control center' here-after) is the center in charge of managing the on ground operations of the payload and data analysis. Whereas the satellite control center manages payload critical operations such as instrument switch on/off and interfaces with the satellite, other payload operations are under the responsibility of the payload ground system:

- Payload operations analysis, preparation, planning and scheduling;
- Payload data processing, archiving and delivery;
- User services;
- Performance analysis and reporting;
- Algorithm tuning and development, verification and validation;
- System maintenance.

A significant number of the functionalities offered by a payload control center remains the same from one mission to the next[1]. In this context, CNES initiated the CMSG project which aims at listing and organizing these common functions to avoid the need to redefine them for each new mission. The objective is to optimize the technical specification and development phase by leaving engineers free to concentrate on the specific aspects of their mission. A functional architecture decomposed into three layers, was then proposed[2]: a data management layer, an application layer and a ground monitoring layer. In the application layer, a function dedicated to the payload health monitoring is identified. This function, as the other ones, must be seen as the addition of a recurrent part and a mission specific part.

In the frame of the CMSG project, the use of generic and reusable components is also promoted to minimize the workload for developing a new payload ground segment and to capitalize the lessons learnt of other missions.

The concomitant need of payload health monitoring for both MSL and SSALTO projects was the opportunity to develop a new generic software component that carries out this function. This software shall help operators, engineers and experts performing the daily monitoring of one or several instruments. To achieve this goal, the software shall automatically warn users when housekeeping telemetry parameters transgress a monitoring rule (like comparison to a threshold, difference with the previous value). The software shall also provide various data displays like a plot of a daily curve, an animated synoptic, etc. Finally, as generic software, it has to provide an extensible architecture allowing the implementation of mission specific requirements such as dedicated user graphical interfaces, specific input/output formats, etc.

B. Mars Science Laboratory (MSL)

Mars Science Laboratory (MSL) is a mission of NASA's Mars Exploration Program, long term effort of the red planet exploration by robots. MSL is a rover designed to assess if Mars has been, or is still, a habitable environment. Its main objectives are thus to determine whether life ever arose on Mars, to characterize the climate and geology of Mars and to prepare for human exploration.

1. General presentation of the rover

To fulfill this mission, the MSL rover, developed by the Jet Propulsion Laboratory, embeds various scientific instruments coming from a worldwide set of laboratories. France is involved in two of the ten instruments. CNES insures the prime contractorship for the French instrument contributions to MSL.

The ChemCam (Chemistry Camera) instrument is jointly developed by LANL and IRAP with support of CNES. The primary objective of ChemCam is to determine the chemical composition of

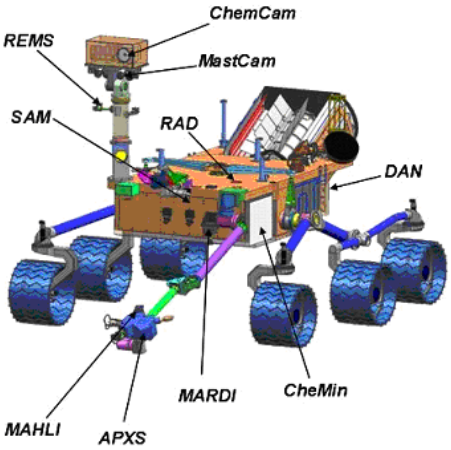

Figure 1. MSL rover

Martian rocks at distances of 1 to 7 meters. It analyzes by spectrometry a plasma light emitted by rocks after a laser shot. A remote micro-imager also provides images of the target.

The SAM (Sample Analysis at Mars) instrument is a suite of 3 scientific units that will perform mineral and atmospheric analyses to detect a wide range of organic components coming from the atmosphere and the ground, from rocks constitutional gases, and search for carbon and other detected atoms isotopes, as well as noble gas isotopes. SAM is under the GSFC responsibility. One of the three units of SAM suite, the "Gas Chromatograph" (SAM-GC), is provided by LATMOS and LISA with support of CNES.

There are two main phases identified for MSL instrument operations. During phase 1, the first 90 sols (about three months), all surface operations are co-located at JPL. ChemCam and SAM are operated by a mixed French and American team. The second phase then begins and will last for a minimum of two years. During this phase, operations are distributed with JPL still serving as a central hub: LANL and IRAP/CNES operate ChemCam on a shared schedule, and GSFC operate SAM suite with support of LATMOS/LISA/CNES.

2. *FIMOC operational requirements*

A dedicated Operation Center is developed and installed in CNES to facilitate the operations from France[3]. The name of this operation center is FIMOC (French Instruments on Mars Operation Center). FIMOC will enable engineers, technicians and French scientists from CNES and laboratories involved in MSL mission, to operate ChemCam and SAM-GC instruments on board the rover during its exploration mission of Gale Crater on Mars.

For ChemCam, the operations are to analyze the every day scientific and housekeeping data and to prepare the work plan for the instrument that follows a very precise activities chronology with meetings in teleconference under JPL responsibility. In order to optimize the resources, the operations will be realized alternatively with the LANL by a period of about 18 days (half of the Earth / Mars time cycle).

For SAM-GC, the operations will be performed depending on the samples collection. But when a sample will be analyzed by SAM, the operations will last several days and the technical and science teams would be mobilized during this time.

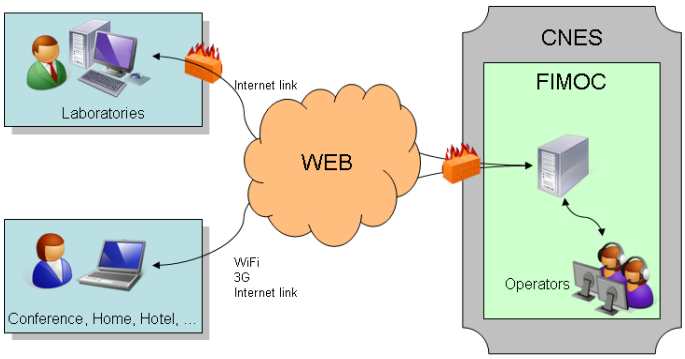

Figure 2. Remote access to the FIMOC

In addition to the instrument supply and science production, each science team also provides support in the rover operations in collaboration with the JPL. The scientists are working remotely to the FIMOC and require remote access to the web server hosting the data products and the monitoring tool allowing to assess the instruments' health. As the operations are daily performed, the scientists will access FIMOC from their office but also everywhere they go (conference, home, hotel, etc.) provided that an Internet connection is available.

In FIMOC, the health monitoring function consists in:

1) Displaying, in engineering and raw values, technologic parameters (temperature, current, voltage, etc.);
2) Checking parameters values with regard to configurable thresholds;
3) Displaying an extrapolation of parameters drifts;
4) Displaying statistics (typically the min, max and mean values of a parameter);
5) Writing reports concerning the behavior and the health of the instrument;
6) Tracking the consumables (e.g. number of shots at certain distance).

Because FIMOC is open to uncontrolled environment (i.e. Internet), security requirements also apply. In order to keep control on the risks taken and to avoid impacting partners or others missions when a security incident occur, a risk analysis has been conducted on FIMOC and the following requirements emerged:

- all input/output data (client/server requests and data files) shall be checked in terms of semantics and syntax before any other processing;
- Systems exposed to the Internet shall be hardened, segmented from each other and isolated by function (database, software application, …);
- System access shall be limited to authenticated users with granted privileges and any authentication attempt shall be logged so that it is possible to trace the originator of any action;
- Systems and software privileges and configurations shall be restricted to the strict minimum.

C. SSALTO

CNES contributes a great deal to ocean observation activities either by developing remote sensing systems (satellites, instruments) or by making data available to scientists. Among the various satellite techniques used for observing the oceans, altimetry is particularly important. The principle is to precisely measure the satellite altitude and then the distance between the satellite and the target surface. By calculating the difference between the two, the 'sea-surface height' is obtained and can be used to deduce a vast amount of information about the ocean and its movements: ocean currents, temperature and salinity variations, tides, etc. Altimetry can also determine wave height and wind speed on the ocean, as well as surface height for lakes and rivers, information about the Earth gravity field, sea ice and polar ice cap topography.

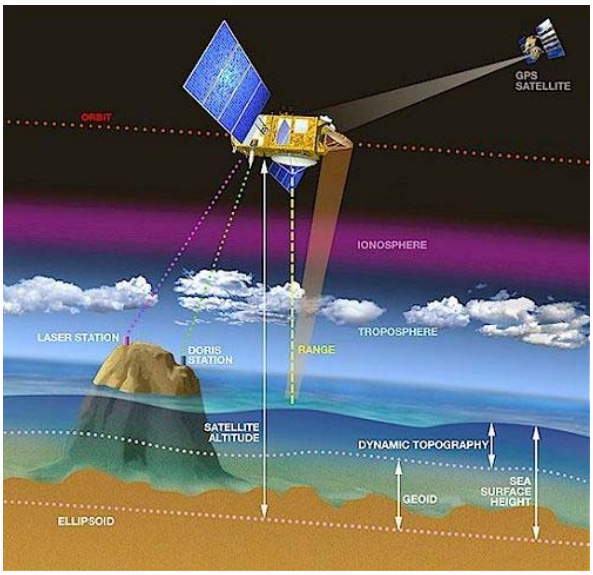

Figure 3. Altimetry measurements principle

With systems such as Doris and GPS, which measure the satellite position extremely accurately, the sea-surface height can be calculated to within just a few centimeters. Doris is also a terrestrial positioning system which can be used to rapidly determine the position (longitude, latitude and altitude) of any point on the Earth surface, to within a few centimeters. This lends itself to numerous geodesy and geophysics applications, such as studies of the movements of tectonic plates or the Earth poles. The Doris system was designed and developed by CNES, the IGN (the French National Geographical Institute) and the GRGS (the Space Geodesy Research Group). Doris instruments are flying on the Spot, Hélios and Pléiades satellites. Since the Topex/Poseidon mission ended, Doris has been a central component of current and future altimetry missions, with the Jason-1 to 3, Envisat, Cryosat-2, HY-2A, Saral, Sentinel-3, etc.

As far as altimetry and Doris are concerned, station keeping operations, data quality assurance, product development and distribution are handled by SALP (CNES Altimetry and Precise Location Department). SALP is responsible for operating the in-flight altimetry and orbitography missions as well as for the development of the payload ground segment for upcoming missions. It operates in an international cooperation frame (NASA, ESA, EUMETSAT, NOAA, ISRO, NSOAS, etc.), with the permanent goal to satisfy a large set of users from scientists to operational users.

1. SSALTO general description

To achieve this goal, CNES has set-up a multi-missions payload control center named SSALTO (payload ground segment for altimetry, orbitography and precise location) which operates the altimetry and Doris missions with the same set of tools. SSALTO performs the processing and distribution of scientific data, and archives the operational data. SSALTO gathers all the payload ground segment facilities required to:

- Command and control the payload instruments (in particular the Doris instruments and the altimeters);
- Control and manage the Doris beacons network;
- Generate precise orbit determination files with different accuracies depending on the production time;
- Generate level-2 altimeter products in near-real time (for operational use like marine meteorology) or offline (for ocean studies);
- User services: archive and distribute products to the user community.

The following missions are currently operated by SSALTO: Doris/Spot2, 4, and 5, Topex/Poséidon, Jason-1 and 2, ENVISAT, CryoSat-2, HY-2A, and in the future SARAL/AltiKa, Sentinel-3A, and Jason-3.

2. Payload health monitoring requirements in SSALTO

With the integration of SARAL/AltiKa and HY-2A in SSALTO, the needs for instruments monitoring increased. Indeed, for these two missions which are respectively realized in cooperation with ISRO and NSOAS, the satellite control center is not located at CNES and thus the health monitoring of the payload instruments can not rely on the usual tools available in the CNES satellite control centers.

Two groups of functions are required to monitor the Doris and altimeter instruments in SSALTO:

1) Routine monitoring which comprises:
    a. Automatic triggering of alarms after telemetry acquisition and monitoring rules checking;
    b. Telemetry parameters display with very synthetic information because of the number of missions and instruments to monitor;
    c. Computing of the distance to the reference ground track: the distance is plotted over a time period to verify if the maneuvers are correctly realized to insure that the ground track does not diverge;
    d. Generation of periodical reports.
2) Expertise monitoring which comprises:
    a. Long term analysis of telemetry parameters;
    b. Location of particular events (such as commands to calibrate the altimeter instruments).

D. On-call operations

The CNES payload control centers are not manly operated 24/7. On-call operations are thus organized to prevent situations where on-board anomalies or ground system failures occur outside office hours. From a technical point of view, these on-call operations are only possible if an automated alarm system warns operators when it detects a problem. The current system at CNES is called SYGALE.

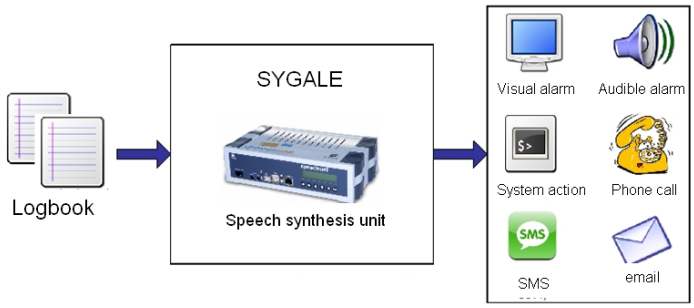

Figure 4. SYGALE general description

In broad terms, SYGALE analyses the content of log files to trigger specific actions. SYGALE can read logs accessible locally or remotely and can also listen to a data stream transmitted over a TCP/IP socket connection. It then checks if any part of the logs, matches a configurable regular expression. Depending on the pattern found, SYGALE computes an alarm message and triggers specific actions for each alarm: sending out simple audio messages (using a speech synthesis unit), sending out voice messages by phone, e-mails or SMS or executing specific system actions (such as software restart).

Even if SYGALE is secure, reliable and effective, it does not allow operators to query the whole context of the anomaly so that they can remotely identify if the alarm message deals with a real anomaly or is linked to an aberration value. For example, if the anomaly is related to a telemetry parameter that is out of range, it may be useful to display the trend of the parameter.

SYGALE does not either allow operators to evaluate the urgency of the anomaly. Only information related to the parameter in the alarm is conveyed. There is no means to get a complete status of the payload and to become aware of orbital events. For instance, it is unnecessary for an operator to go to the payload control center in a hurry in the middle of the night if the next two orbits are blind and then the command to “repair” the anomaly can only be sent in 6 hours.

One of the objective of the IMIS with regard to on-call operations, is to provide a more comprehensive means to remotely perform the first level monitoring and diagnostics: plot the values of one or several parameters over a period of time, receive new alarms and eventually acknowledge them, create a message describing an alarm context with plots or free texts and send it to a third party (for example the expert of an on-board instrument), take note of the operations schedule, etc.

The need for mobility must also be taken into account when defining a tool for on-call operations. Indeed, by definition, an operator that is on call should not have to stay at home with a computer physically connected to the Internet but he should remain contactable wherever he/she goes. For these reasons, an on-call application must be available on a mobile platform.

Fortunately, recent technological improvements can now fulfill these needs. On the one hand, the mobile network offers the bandwidth to transfer large amount of data and not only voice communication. On the other hand, Smartphones have much greater resources to run applications that become rich: CPU (e.g. 1GHz) and RAM (up to 512 KB or even 1024 KB on the most recent mobile platforms). Finally, new concepts of human – computer interaction (the now well-known ‘multitouch’ interaction first introduced on mobile devices by the Apple iPhone) provide new opportunities to develop applications that display content even if the screen is small (e.g. 800x480 pixels on a 4 inches screen).

III. IMIS solutions

The Instrument Monitoring Interactive Software (IMIS) main goal is to allow the operator of a scientific payload control center to monitor the payload instruments' health on a daily or weekly basis. The instruments' health is monitored through the telemetry parameters retrieved from the satellite control center. In some cases, it is also relevant to correlate the monitoring of telemetry parameters with the commands that were executed on-board: for example, when monitoring a parameter indicating the instrument functioning mode, it may be interesting to know when commands modifying this mode were sent. Moreover, even if telemetry parameters and commands are temporal data, it may be useful to identify the position of the spacecraft where an event (execution of a command or alarm on a telemetry parameter) occurred. This provides a different outlook on the instrument(s) health thanks to the correlation of the monitored data with spatial information that can not be directly made with a 'time-based' monitoring (for example, analysis of the South Atlantic Anomaly effect on the instrument). Finally, for long-term monitoring, it may be more convenient to monitor the trend of a parameter instead of all its individual values. Indeed, it enables to highlight the global evolution of the parameter (for example, identification of parameter drifts) and to hide aberration values, high-frequency phenomenon, etc.

For this purpose, IMIS software handles three main categories of data:

1) Telemetry parameters comprising parameters measured on-board the spacecraft and ground-computed parameters: these parameters are the basis of the monitoring function of IMIS;
2) Executed commands: this is the list of commands uplinked to the spacecraft and executed on-board;
3) Ephemeris data enabling to correlate the temporal telemetry parameters with the spacecraft location.

It is under the payload control center responsibility to regularly supply IMIS software with these data. In particular, the payload control center is responsible for decommuting the telemetry (comprising the computation of engineering values and significance statuses), identifying telemetry holes, processing ground-computed parameters and finally sending one or several files with all these data to IMIS.

The data lifecycle in IMIS software is the following:

- Data are acquired by IMIS software which computes the monitoring statuses and raises the corresponding alarms. Acquired data and additional computed information are stored by IMIS in an appropriate manner.
- For trend analysis purpose, a sampling of the acquired telemetry parameters can be achieved. There are different ways to sample the parameters: the sampled values can be the minimum, maximum and/or mean values over the sampling period.
- A purge process can be scheduled to erase the oldest telemetry parameters. The objective is to ensure disk capacity and high performance when requesting data (because there are less data to request). The activation policy of the purge process depends on the mission characteristics (in particular the volume of telemetry parameters produced during the mission lifetime).

The design of IMIS was meant to allow users from anywhere (but of course with sufficient privileges) to access the monitoring information. It results in classical 3-tier architecture: the server acts as a data provider whereas the clients display the data in graphical user interfaces. Service oriented architecture is set-up to implement the client / server communication using the Java RMI technology.

Two kinds of IMIS clients are available. The first one is a desktop application (compatible with Windows, MacOS and Linux operating systems) which communicates with the server through the Internet. The second one called SmartMonitorer is a light 'IMIS' client for any Android platform (smartphone or tablet) connected to the Internet via a Wi-Fi connection or the mobile broadband network (for example, 3G network).

A. IMIS desktop application

IMIS software is designed to provide users access to the monitoring data. It provides several functions. The first one is the ability to configure the monitoring by setting-up the list of parameters that shall be monitored, defining the monitoring rules applied on parameters (i.e. the condition when a parameter is in alarm or in nominal state) and configuring the trend-analysis and purge mechanism.

The software then offers several ways to represent and analyze the data:

- **Line charts** plot one or several parameters over a period of time. Line charts allow time series and XY series (i.e. plot of parameters in function of another). Among the interactive and useful functions available, the most relevant ones are the following. The tangent line and its corresponding slope can be displayed at any point selected on the chart. The polynomial regression can be computed and displayed. The occurrences of commands can be highlighted so that it is possible to correlate the evolution of a parameter with the execution of commands.

- **Histogram charts** plot the distribution of the values. It allows to identify aberration values, to verify how many values are near, for example, their upper bound, etc. It gives a statistical point of view.
- **Synoptic** displays the values and statuses of a set of parameters at a given time. It is a kind of snapshot providing the status of a subsystem.
- **Dashboards** enable users to get synthetic information about the monitoring state of a set of parameters. A dashboard displays their current values, their current monitoring states and an indicator of whether unacknowledged alarms related to the parameters have been raised.
- **Planisphere** displays a user-friendly global map, including a basemap containing static geophysical elements (continents, inland seas …) and some geo-referenced objects (satellite ground tracks, station visibility circle). Along the satellite ground track, it is possible to highlight the alarm occurrences and to display the variation of the parameter values.

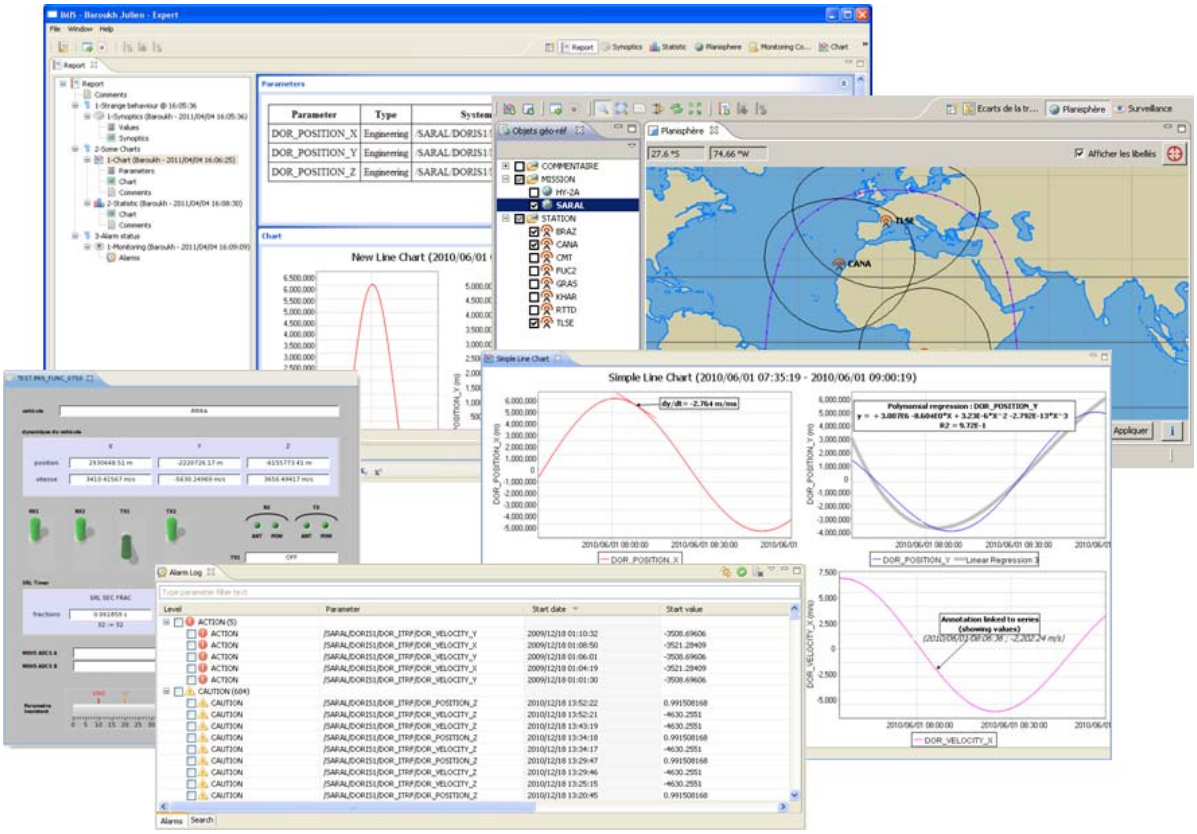

Figure 5. IMIS desktop application

In case a parameter transgresses a monitoring rule, an alarm is generated and displayed in a dedicated logbook. Once the alarm cause has been identified, users can acknowledge the alarm (pending they have rights to do so). Finally, users can annotate and comment all these displays with valuable information of their choice. If need be, the results of the users analysis can be assembled in a report to publish.

The IMIS desktop application was designed to be highly interactive, intuitive and configurable. There are several links between displays so that it is possible to watch the same event with different point of view: for example, when plotting the values of a parameter, it is possible to display the histogram chart of the same parameter on the same period of time.

B. SmartMonitorer

SmartMonitorer is an application that runs on any Android compatible platforms whether a smartphone or a tablet. It communicates with IMIS to retrieve the information it displays.

Figure 6. SmartMonitorer Android application

SmartMonitorer is mainly designed for on-call operations and then only contains a subset of the IMIS functionalities. The most important function with regard to on-call operations is the management of alarms. Thanks to the native notification system available on Android, SmartMonitorer can warn users when a new alarm is detected. The phone (or tablet) vibrates and emits a beep until the user views the alarm. Users can then display any details about the alarm (start date, end date, telemetry parameter in alarm, monitoring rule that is transgressed, etc.). When the cause of the alarm is identified, users can acknowledge it.

In order to help users analyzing the alarm, SmartMonitorer provides a very convenient and user-friendly function: the plot of telemetry parameters over a configurable period of time. Not only can users plot the parameter in alarm, but up to four parameters can be displayed simultaneously on the same graph or on separate graphs. The list of available parameters is defined in the satellite database that is retrieved from IMIS.

Finally, thanks to the native SMS and mail applications, SmartMonitorer can send messages with valuable information about the alarm: text, alarm details, screenshots of charts. Other features are foreseen but not yet implemented such as the display of the satellite ground track using the Google Map application provided with the Android system.

IV. Design

A. IMIS client / server architecture

The IMIS client / server architecture is a Service Oriented Architecture. It ensures flexibility (services are independent of each other) and adaptability to changes thanks to the clear separation between the business logic and the ‘services’ interface layer. New services can then be implemented to meet specific requirements of a space mission without strong modification of the IMIS architecture or impact on other missions.

In IMIS software, the communication layer is implemented by a component named ‘channel’. The channel defines the services with their input and output data (i.e. the API of the communication layer). The channel also provides an implementation of the communication layer that is described here-after.

Different modes of interactions between the clients and the server are implemented. With **asynchronous services**, the service invocation is not blocking. This interaction mode is extended to deal with operations such as ‘Progress’ enabling to continuously receive the results as work progresses on the server side. **Synchronous services** are characterized by the client invoking a service and then waiting for a response to the request. Because the client suspends its own processing after making its service request, synchronous services are suited when the service can process the request in a short time or when a small amount of data is returned. Finally, a last interaction mode is implemented to manage **notifications** from the server to one or more clients. The clients first subscribe to one or several events. When such an event occurs, the server notifies the client.

These three interaction modes highlight the exchange of data that are initiated by the client (synchronous and asynchronous calls) or by the server (asynchronous call-backs and notifications). However, the technical solution must also take into account the following security constraints: the link between the clients and the server shall be able to pass through firewalls that only allow HTTP connection and the communication shall be initiated by the client.

As a consequence, the channel is implemented using RPC (Remote Procedure Call) paradigm with two transport-layer protocols:

- Java RMI (Remote Method Invocation): this implementation is convenient for IMIS deployment in an Intranet that allows the transfer of Java serialized objects;

- HTTP-Tunneling RMI: the Java serialized objects are encapsulated in an HTTP request / response so that they can be transferred over the Internet.

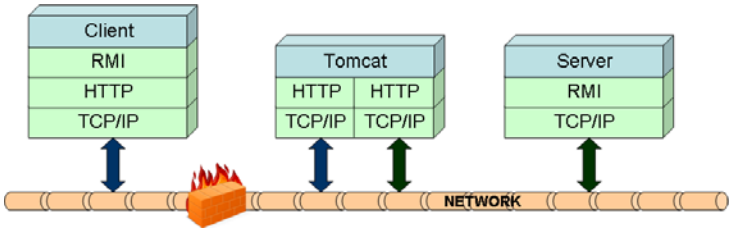

Figure 7. HTTP tunnelling

The principle of the HTTP tunnel consists in inserting an HTTP relay to route the RMI traffic via HTTP POST requests. The relay is a Java Servlet container (e.g. Apache Tomcat) that receives RMI requests and transfers them to the RMI server. A variant of this approach is to integrate RMI services directly in the Servlet container in order to improve performance on the encoding and network traffic.

Finally, because only the clients are able to initiate the communication, the notification mechanism used for example to notify clients when new alarms are raised, is the following: the client frequently queries the server to know if any new event has occurred.

B. Client / server architecture for SmartMonitorer

SmartMonitorer is an Android application written in Java. Note that in the Android platform, there is no Java Virtual Machine: Java classes are compiled into Dalvik executables and run on Dalvik which is a specialized virtual machine designed specifically for Android. Unfortunately, all the Java API is not supported by Dalvik, in particular Java RMI can not be used.

Another transport layer has then been set-up to make the communication between SmartMonitorer and the IMIS server possible. The services are implemented using HTTP and the principles of REST (Representational State Transfer). REST is a style of software architecture for distributed systems in which all services are defined as resources that can be accessed via HTTP methods (GET, PUT, POST and DELETE). The resources are thus identified by their URI (which enables to access the service) and the internet media type of the data transferred (e.g. XML, JSON). SmartMonitorer uses the JSON format which is less verbose than XML. This is a very important criterion because data are transferred through the mobile network that has a limited bandwidth (in theory 2 Mbits/s, but in practice around 400 Kbits/s).

Moreover, in REST architectures, the services are stateless between requests. It means that the services do not carry any session state on behalf of the clients that are not currently in the process of making requests. All information about the session state must then be transmitted each time a request is made. One example of this point in SmartMonitorer is the session ID that is provided when a user is connected (with its login / password). The session ID is first retrieved by calling the authentication service and then transferred as a parameter of all subsequent requests (such as the retrieval of telemetry values).

Another point that is not directly related to REST architecture is the way SmartMonitorer is notified when new alarms occur. Even if Google provides a framework named Cloud to Device Messaging (C2DM) to achieve this goal, SmartMonitorer does not use it. On the one hand, for satellite on-call operations, CNES does not want to be dependent of a third-party that is not clearly committed to a certain level of quality of service. On the other hand, CNES does not want to provide any third-party with information about the monitoring and health status of the satellites it operates. The notification mechanism that is implemented is thus a frequent query of the server to know if any new alarm has raised. The frequency can be tuned to avoid a huge number of connections limiting the available bandwidth and the battery life.

In concrete terms, the client / server architecture of SmartMonitorer simply adds a new layer to the architecture described in figure 7. This layer is named 'IMIS gateway'. It is composed of several RESTful services that simply handle the HTTP requests of SmartMonitorer and invoke the RMI services. The HTTP parameters in JSON format are also converted in Java objects. Finally, the results provided by the RMI services are forwarded to SmartMonitorer in the JSON format.

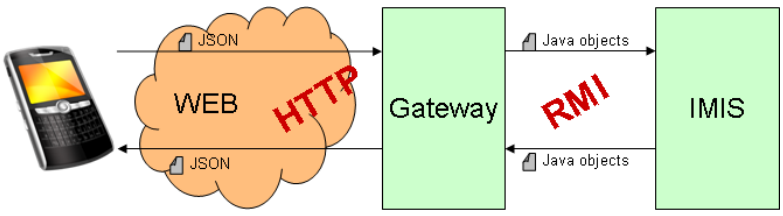

Figure 8. SmartMonitorer client / server architecture

C. Technologies

Java is the programming language selected for developing IMIS because of the IMIS distributed architecture and the complexity of the IMIS GUI. Java applications can run on any Java Virtual Machine regardless of the computer architecture. Java is also the language for developing on Android platform. Finally Java development environment offers a great number of APIs (Application Programming Interface), IDE (Integrated Development Environment), frameworks and a continuous integration environment. One particular API that is used in IMIS is **Java RMI** (Remote Method Invocation) which enables to create distributed Java technology-based applications, in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. Java RMI is the technology used to implement the IMIS channel.

PostgreSQL is the object-relational database management system used to store all persistent information of IMIS except telemetry parameters values which, for performance reasons, are stored in binary files using a CNES proprietary technology named "**TelemetryTable**". The principle of the "TelemetryTable" is to store the telemetry covering a specified period (typically one day) in a binary file with an extensible time index. Each file is self sufficient in the sense that it includes a description of its own data dictionary necessary to read / write the data blocks. The advantages of this binary structure that is temporally indexed are its minimum disk space footprint (data are not encoded in ASCII) and the optimal performance for inserting and retrieving data on a time criterion.

It remains to present the technology best suited to implement the GUI of IMIS desktop application. Swing comes to mind as this technology is an integral part of the standard Java API. Nevertheless there are interesting alternatives, including SWT (Standard Widget Toolkit) and more specifically the **EclipseRCP** technology (based on SWT and JFace). EclipseRCP technology enables developers to implement with a good productivity, highly interactive and user-friendly applications. Indeed, the Eclipse framework provides a very mature architecture with reusable components and standard user interactions (such as drag and drop, windows docking, online help, etc.). The architecture is based on plug-ins that can be made available gradually throughout the development phase. Moreover, as a generic tool, IMIS has to take into account the specificities of every mission it operates and the modular architecture of EclipseRCP allows to add or remove plug-ins depending on the user requirements.

To illustrate the architecture based on reusable components, we can mention the integration of four components in the IMIS client:

1) **BIRT** is an open source Eclipse-based reporting system that IMIS uses to produce health reports.
2) **CSS** is an integrated control system based on the Eclipse environment (http://css.desy.de). IMIS uses CSS to display and animate synoptic.
3) **uDig** is an open source desktop application framework built with EclipseRCP. The goal of uDig is to provide a complete Java solution for desktop GIS data access, editing, and viewing. uDig is used as a plug-in in IMIS to display a map with the satellite ground tracks and stations.
4) **JFreeChart** is a free Java chart library that makes it easy for developers to display professional quality charts in their applications. JFreeChart is compatible with both Swing and SWT making it possible to integrate into an EclipseRCP application.

V. Conclusion

IMIS development ended in November 2011. Since late 2011, IMIS has been deployed in the MSL payload control center and has been integrated into SSALTO. Operators, science teams, and experts from CNES and French/American laboratories involved in the MSL project have a very good feeling about the software. IMIS meet the requirements of two different missions (Earth observation and Mars exploration) that at a first glance are supposed to have distinct operational procedures to monitor the instruments. This was possible because IMIS was designed as generic software that provides the base functionalities and an extensible architecture. TARANIS and SWOM are the next missions that intend to use IMIS in their payload control centers[4].

Technologies and software design concepts are now mature and allow to implement distributed Web architecture. In terms of information system security, no specific difficulty was encountered. The lessons learnt are to consider upstream the security constraints. In most of cases it is sufficient to apply the state of the art in matter of security: network segmentation and isolation, system hardening, and application security (use of security guidelines and best practices from CNES and partners: authentication, traces and security logs, restricted privileges, input/output data checking, etc.).

With regard to SmartMonitorer, it was a good opportunity to check the capabilities of mobile platforms to develop a professional application and to validate the operators' interest in a mobile application that allows to remotely monitor the payload instruments. SmartMonitorer has raised a lot of interest and its use is now being considered for the MSL mission. SmartMonitorer offers new functionalities and possibilities for on-call operations

that have been requested by operators for a long time. However, as it is not currently used in an operational context, there is currently no lesson learnt about the real advantage of using such a tool. Moreover, the deployment of SmartMonitorer requires further analyses: definition of security procedures relative to professional usage of Smartphones (for example: capability to remotely delete data and application when the device is stolen), modification of the operational procedures because the way on-call operations are realized, is modified with such an interactive tool that provides means to remotely analyze and acknowledge alarms, etc.

As far as technical aspects are concerned, on the one hand all features necessary to the SmartMonitorer development were available and well-functioning on the Android platform. But on the other hand, it should be noted that the development faced a rapid evolution of the Android operating system (two major releases within 6 months). Moreover no mature chart API was found even though a beta version of a library named aFreeChart was used successfully and works pretty well.

Appendix A
Acronym List

API	Application Programming Interface
ChemCam	Chemistry Camera
CNES	Centre National d'Etudes Spatiales (French Space Agency)
CMSG	Centre de Mission Scientifique Générique (Generic Scientific Payload Control Center)
DORIS	Détermination d'Orbite et Radiopositionnement Intégré par Satellite
ESA	European Space Agency
EUMETSAT	EUropean METeorologic SATellites
FIMOC	French Instruments on Mars Operation Center
GIS	Geographic Information System
GPS	Global Positioning System
GRGS	Groupe de Recherche de Géodésie Spatiale
GSFC	Goddard Space Flight Center
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
IGN	Institut national de l'information géographique et forestière
IMIS	Instrument Monitoring Interactive Software
IRAP	Institut de Recherche en Astrophysique et Planétologie
ISRO	Indian Space Research Organisation
JPL	Jet Propulsion Laboratory
JSON	Java Script Object Notation
LANL	Los Alamos National Laboratory
LATMOS	Laboratoire Atmosphères, Milieux, Observations Spatiales
LISA	Laboratoire Interuniversitaire des Systèmes Atmosphériques
MSL	Mars Science Laboratory
NASA	National Aeronautics and Space Administration
NOAA	National Oceanic and Atmospheric Administration
PI	Principal Investigator
REST	REpresentational State Transfer
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SALP	Service d'Altimétrie et de Localisation Précise
SAM	Sample Analysis at Mars
SAM-GC	Sample Analysis at Mars - Gas Chromatograph
SMS	Short Message Service
SOA	Service Oriented Architecture

SSALTO	Segment Sol d'ALTimétrie, d'Orbitographie et de localisation précise
SWT	Standard Widget Toolkit
NSOAS	National Satellite Ocean Application Service
XML	eXtensible Markup Language

Appendix B
Glossary

sol	The term sol is used by planetary astronomers to refer to the duration of a solar day on Mars (source Wikipedia)

Acknowledgments

Authors would like to thank all the team involved in the financing and the development of IMIS and SmartMonitorer for their confidence, support and motivated contribution: CNES, LANL, IRAP, LATMOS, LISA and Thalès company (in charge of the development of IMIS).

References

Proceedings

[1]Pradels G. and Guinle T., "Generic mission centers for small satellites", 4S Symposium, 2006.

[2]Pradels G., Baroukh J., Queyrut O., Sellé A. and Malapert J.C., "CNES solutions for a reusable payload ground segment", IAC-11.B4.3.6, 2011

[3]Lafaille V. et al, "A CNES remote operations center for the MSL ChemCam instrument", SpaceOps, 2010

Unpublished Papers and Books

[4]Camus A.L. and Pradels G., "Reusable toolset for an Easy-to Build Payload Ground Segment", SpaceOps, 2012