TELMA
An Evolutive Distributed Application for In-Orbit Satellites Support

Patrick Pleczon, Eva Bonnin and Laurent Desplas
Astrium, 31 rue des Cosmonautes, Toulouse, France

Keywords: Satellites In-Orbit Support, Distributed software, Scalable application, Evolutive application.

Abstract: Satellites In-Orbit Support is a complex activity requiring the daily thorough analysis of thousands of parameters. Performing this activity for the growing fleet of satellites delivered by Astrium, the space EADS subsidiary and leading European Satellite company, featured the development of a new generation processing environment: TELMA (standing for TELeMetry Access).
This paper describes TELMA and provides a detailed view of the technical choices that structured this system. First, it describes the notion of satellite telemetry and the needs of the In-Orbit Support (IOS) team. Then TELMA software architecture is detailed as well as system deployment aspects. This paper also provides lessons learnt about the development and deployment of the software. Finally, a status of the current version of TELMA is presented and perspectives are drawn.

1 INTRODUCTION

Satellites In-Orbit Support is a complex activity requiring the day-to-day thorough analysis of thousands of parameters. Performing this activity for the growing fleet of satellites delivered by Astrium, the space EADS subsidiary and leading European Satellite company, dictated the development of a new generation processing environment: TELMA (standing for TELemetry Access). This paper describes TELMA and provides a detailed view of the technical choices that structured this system.

2 SATELLITE TELEMETRY

Satellite telemetry allows monitoring satellite components. Geostationary satellites continuously send a telemetry flow that is received on ground stations. Telemetry is generally emitted as a binary stream encapsulated within a dedicated protocol. A single telemetry consists in an on-board measure of a satellite parameter (e.g. component pressure, voltage, on board software memory dump, etc.). It can be coded on any number of bits between 1 and 48 depending on the data type. Such a value received on ground is called a raw value. The activity of processing this telemetry flow to restore time stamped engineer (i.e. physical) values is called "telemetry decommutation". It contains at least the following steps:

- Extracting the raw value from the binary flow.
- Calculating the data time stamp.
- Applying an optional calibration function to restore the physical value.
- Applying an optional scale factor.
- Computing a parameter-dedicated algorithm that establishes the parameter validity according to the values of other decommuted parameters.

3 IN-ORBIT SUPPORT NEEDS

In-Orbit Support team is in charge of the analysis of the satellite state during its whole lifetime (around 15 years) in order to keep improving satellite quality and support customers in managing satellite configuration. Specialists of satellite sub-systems (e.g. propulsion, power generation, etc.) have to monitor their sub-systems daily by examining the behaviour of a set of satellite parameters. In some cases, they have to investigate the long term behaviour of a sub-system by analysing the past satellite telemetry, even from the beginning of the satellite life.

The specialists' first need is therefore to display various graphical representations of parameters:

Pleczon P., Bonnin E. and Desplas L..
TELMA - An Evolutive Distributed Application for In-Orbit Satellites Support.
DOI: 10.5220/0003445101990202
In *Proceedings of the 6th International Conference on Software and Database Technologies* (ICSOFT-2011), pages 199-202
ISBN: 978-989-8425-76-8

- Plots tracing hourly or daily statistics about one or several analogue parameters.
- Daily plots that can be used to focus on any value of one or several related parameters.
- Statuses plots (related to telemetries with discrete values such as On/Off, etc.) that shows changes of parameters values.
- Comparisons of parameters evolution of different satellites.

They also need to check daily the results of conditions that can be expressed by a mathematical expression using parameters values. In some cases, they have to elaborate more complex processing that cannot be managed with simple monitoring function in order to produce data or specific plots.

Users have also to generate reports in different formats (Microsoft doc, PDF, etc.). It is aimed to give a clear status of satellites state, in terms of performance, lifetime, irregular events, anomaly predictions, etc. These documents contain data from many sources (raw telemetry, statistics, processing output, comments, etc.). Reports can cover various periods of time and can address one sub-system or all sub-systems of a satellite.

In addition to these basic functional needs, these following requirements are also necessary:

- Configurability: possibility for users to define several views including a set of related plots.
- Performance: the users need to be able to get their data at the beginning of their working day. This requires TELMA being able to process satellite telemetry of the day before, for the whole fleet, in less than 8 hours.
- Reliability: although this is not a critical real time system, TELMA shall minimize the effects of hardware or software problems.
- Reprocessing: TELMA shall allow the reprocessing of past data on long periods.
- Hardware: the customer requirements are to be able to use regular desktop PCs with standard company configurations as processing units, and to be able to access to TELMA from their office.
- Scalability: the initial configuration shall support 32 satellites and shall be scalable to add new satellites regularly, up to 60 satellites. TELMA shall also be scalable to add new satellite parameters and new monitoring and processing.

4 TELMA MAIN FUNCTIONS

TELMA answer to fulfil users' needs rely on the daily production of user-defined set of plots starting from telemetry decommutation. Users can define groups of related plots they can consult at a glance. Selecting a given plot displays an interactive plot that can provide detailed values. Figure 1 shows a sample of statistics and interactive plots.

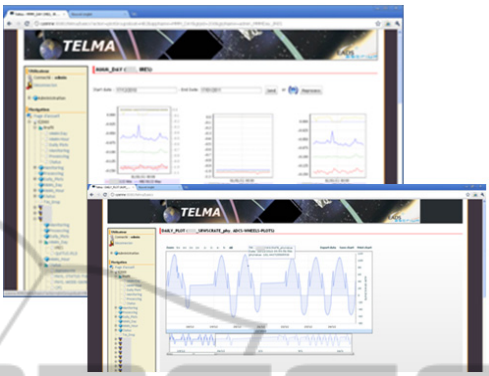

Figure 1: Statistics and interactive plots.

Monitoring applications are computation procedures executing rather simple expressions using a set of telemetries or data produced by a processing application. They are executed at regular intervals by TELMA in order to assess a conditional expression and to execute the associated actions if the conditional expression is true (such as sending an alarm to a group of users). Monitoring applications are TELMA answer to users' needs for "simple" situation assessment.

For more complex analysis, processing applications are used. They are more complex procedures using a set of telemetries and/or results of other processing to produce data or graphics.

Monitoring and processing applications can be defined and integrated into TELMA by the users themselves. They can be allocated to one or several satellites.

5 TELMA ARCHITECTURE

5.1 Functional Overview

TELMA is composed of the following components:

- A scheduler facility used to distribute the work to be performed to a set of Processing Units (PU) at the right time.
- Processing Units are the "workers" entities.
- A web server provides the graphical user interface to TELMA users.
- A database stores TELMA configuration data, statistics and processing results.

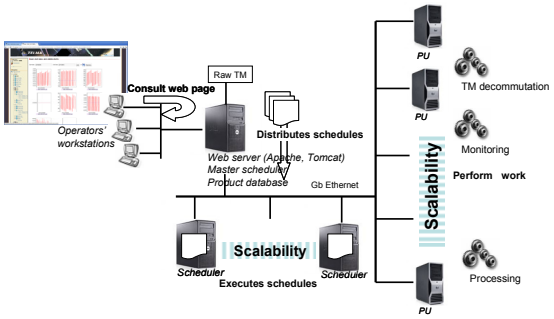

Figure 2: TELMA overview.

TELMA requires a **raw telemetry repository** where satellite telemetries generated by the Satellite Control Centre are stored. Physical telemetry is stored in a physical telemetry cache.

5.2 Major Technical Choices

5.2.1 Languages and Technologies

The non-web parts of the system are developed in Java. The user interface relies on Java Servlets deployed on a Tomcat server, JSP and JavaScript. Flex is used for implementing the interactive plots.

We tried to limit as much as possible the use of third party software to reduce long term maintenance issues.

5.2.2 Decommutation

During satellite design and until its end of life, a system database is maintained by satellite specialists. Among many data, this database contains the extensive definition of satellite telemetry (organisation of parameters in the downlink frames, calibration curves, etc.). There is at least one different database per satellite and generally many versions are produced during the satellite lifetime.

TELMA integrates a satellite database to Java code compiler that generates the decommutation code for each satellite. The code is optimized to avoid table lookup. For instance, algorithms evaluating parameters validity are generated functions using direct Java references on Java objects that implement other parameters.

Individual JARs are generated for each [satellite, database version] pair. JARs are dynamically loaded from a central repository by the PUs when an activity requests a specific satellite decommutation.

Decommutation requests are sent to the scheduler that can possibly split them according to various criteria. Split requests are sent to several PUs and results are merged by the scheduler on PUs answers.

5.2.3 Management of User-defined Applications

Monitoring and processing applications are defined within TELMA user interface using a sub-set of Java and a set of predefined user-oriented functions (such as telemetry access functions, dedicated database access functions). They are compiled and packaged into individual JARs. As for decommutation JARs, the monitoring and processing JARs are dynamically loaded by PUs when they are required.

5.2.4 Processing Units

PUs are simple Java server applications offering an activity control interface (start/end activity, status/load request). A new thread is created for each activity and JARs implementing decommutations, monitoring or processing applications are loaded when needed.

PUs are deployed as a launch script that invokes a JNLP (Java Network Launching Protocol) file. Thanks to JNLP the last version of the code is loaded and run on the PU host each time the PU is restarted.

5.2.5 Scheduler Facility

An initial trade-off has been made between using an existing component or product or designing a dedicated scheduler. We finally chose this later solution to be independent of third party software and to optimize the solution to our specific needs.

The master scheduler is quite simple and its role is only to split the initial schedule (addressing the full fleet) in sub-schedules using configurable criteria, and to distribute the schedules to the schedulers.

Schedulers are able to split incoming requests and request activities to PUs. Currently the scheduler is only time-based. A future evolution could be to add task dependencies.

The main difficulty is to handle the various possible anomalies to ensure that the requested work is actually performed within an acceptable time. This is enforced by several mechanisms such as the use of start and end time-outs for managing PUs errors and a scheduler persistent history to be able to resume the scheduler state in case of scheduler crash.

5.2.6 Inter-processes Communication

Inter-processes communication relies on RMI (Java Remote Method Invocation). RMI interfaces remain quite simple and are basically commands and control functions and notifications.

The most complex interface is the one dedicated to the telemetry decommutation. A distributed callback pattern is used to avoid blocking on the decommutation request. Data are returned to the caller at the end of the decommutation as a structured collection of proxies on the actual telemetry data that are stored in the physical telemetry cache by the PU.

Statistics and processing data are directly stored in the database as there is no need to put a middle tier for this that would just have added additional delay to store data.

The PUs regularly send load information to the schedulers so that they can select the best PU for starting a new activity. As we have already mentioned, time-out are used to detect PU malfunctioning and to possibly restart the applications started on such a PU on another one.

5.3 System Deployment

TELMA deployment is very versatile. Everything can fit on one single server if a small system is required for one satellite, or tens of PCs and several servers can be used for an operator that has to manage a large fleet of satellites. Scalability is possible at different level. First of all for processing units that form the active part of TELMA. Secondly, Schedulers can be added to support very large schedules. In addition to this, several Tomcat servers can be used with load balancing.
The current platform is based on 4 servers connected to a SAN, 15 PC and manages data produced by 32 satellites.

6 LESSONS LEARNT

One of the main issues in web-based graphical user interfaces development for an application that has to be maintained during years is the proliferation of tools, technologies, libraries and languages, questionable maturity and severe long-term maintenance issues. Developing everything with basic components can be expensive, especially building advanced interactive components. Using a third party library or toolkit can be a good solution but the durability of the solution shall be questioned. Some technologies or toolkits analysed at the very beginning of the project have already disappeared!

In addition, we encountered several robustness issues due to third party software. Especially we ran into some crashes of the Java Virtual Machine with weird “out of swap space” errors although we had a lot of free memory. Most of the issues were solved by upgrading the faulty software with more recent versions.

7 CURRENT STATUS AND PERSPECTIVES

The current version of TELMA is used every day on Astrium fleet. Performance is correct as well as robustness.

Several improvements are already planned from details adjustments to major evolutions. Among these major evolutions, we can quote:

- The implementation of a graphical user interface for monitoring and processing definition based on a graphical representation of algorithms.
- Near real time monitoring in order to perform monitoring and processing activities on the last x minutes of telemetry.
- The split of other types of requests that would speed-up reprocessing on long periods.

ACKNOWLEDGEMENTS

The authors would like to thank Astrium IOS team for its fruitful collaboration in the definition of TELMA and its involvement for improving this product.

REFERENCES

Adobe Flex, accessed 2010, adobe.com/products/flex/.

Apache Software Foundation, accessed 2010, The Apache Tomcat 5.5 Servlet/JSP Container, tomcat.apache.org/tomcat-5.5-doc/index.html.

Berstis, V., 2002. Fundamentals of Grid Computing, *IBM Red Books*.

Dojo Toolkit Community, accessed 2010, Dojo Toolkit Reference Guide, dojotoolkit.org/reference-guide/.

Eadline, D., The State of Oracle/Sun Grid Engine, Sept. 2010, Linux Magazine.

Schmidt, D.C., Stal, M., Rohnert, H., Buschmann, F., Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects, 2000, Wiley & Sons.