A WEB-BASED MODULAR AND FLEXIBLE DATA ACQUISITION AND TELEMETRY MONITORING SYSTEM FOR MICRO SATELLITES

Mehran Sarkarati
ESA/ESTEC, SCI-SB, Keplerlaan 1, NL-2201 AZ Noordwijk ZH
Tel: (31) 71 565 8391, Email: mehran.sarkarati@esa.int

1. ABSTRACT

The process of software development for the ground segment of micro satellites is a very time consuming and costly procedure. Most of the micro satellite projects have come up, therefore, with their own specific, non standard software solutions, which are generally customized to the specific project requirements and hence not reusable in other projects. In this paper a generic, flexible and reusable software architecture and a corresponding application programming interface (API) are introduced, which can reduce the time and costs of the software developments significantly for new missions.

2. INTRODUCTION

Micro satellites, often built in low budget projects by universities and research organizations, are very rich in the variety of their concept and design. Their EGSE (Electrical Ground Support Equipment) vary consequently with respect to the details of their space-ground data link and the available ground station facilities. Hence the developers of micro satellites are in general forced to implement completely new EGSE software, which is responsive to the needs of their own satellite, mostly due to the lack of on-the-shelf software.

Although many attempts have been undertaken and a number of standards such as CCSDS or ECSS have been developed which are applied by most of the space agencies, micro satellite projects usually renounce using them to avoid additional system complexity. To address this issue a comprehensive, reusable and extendable programming library and a well-designed modular software architecture have been developed, which will be described in detail in this paper.

3. THE GENERIC CONCEPT OF TELEMETRY DATA HANDLING

The concept of the telemetry data handling is essentially the same for all micro satellites and can be described in the following simplified procedure:

1. Receive the data stream from the satellite
2. Extract the individual TM frames out of the data stream.
3. Extract the raw value of every single TM data from the Frame.
4. Apply the calibration routines to retrieve the engineering value of the TM data.
5. Visualize the values to the clients through some User Interface.
6. Archive the retrieved TM data.

Based on this generic procedure the system architecture shown in figure 1 has been designed, which guarantees the highest level of flexibility and extendibility for different satellite projects. The system can be divided in the following core software components:

- The TM database, which contains the definitions of all telemetry data. These definitions consist of information about the raw format of the data, i.e. the size in Bytes and Bits, its position in the TM frame, the calibration method to retrieve the engineering value and hard/soft limits for valid values, etc.
- The data handling engine, which establishes a connection to the database and to the satellite receiver hardware at the ground station to carry out the task Nr 3 and 4 of the above procedure. Upon a satellite pass it receives telemetry packages sequentially and extracts the telemetry data entities.
- Housekeeping monitoring and other telemetry clients to visualize, archive and process the extracted telemetry data (Tasks Nr. 5 and 6).

![Figure 1: The Architecture of the generic telemetry data handling system](image)

It has been a conscious decision not to include the tasks 1 and 2 in this software architecture, since there are already a variety of commercial, on-the-shelf hardware available, such as Bit- and Frame-Synchronizers, which perform these low level tasks.
4. MODULARITY AND FLEXIBILITY OF THE APPLIED SOFTWARE DESIGN

Analyzing the introduced generic concept, the following critical interfaces have been identified, which may alter in different satellite projects and need hence to be flexible.

- Connectivity to the ground station hardware
- TM frame structures and applied communication protocols
- Information needed to define the TM data entities in the database
- Requirements for Telemetry Clients

A short case study shows that it is not a realistic approach to design and offer a ready-to-use software packet, which should be responsive to the very different and sometimes contradictory requirements of different satellite missions, by setting up a few number of editable configuration options. It has, therefore, been decided to design a modular and extendible software architecture and to offer an API, which enables the software engineers to extend the system and adapt it to their project requirements.

The Flexibility of the system has been guaranteed by using excessively the concept of realization relationship (Behavior Agreement) between different components of the system. This has been realized by defining a number of Java Interfaces, which cover the above mentioned critical arias of the system. Since Java Interfaces are classes, which define and dictate a certain behavior, without providing a concrete implementation, satellite engineers can provide their own implementations of that behavior with respect to their project requirements.

Using this “plug and play” approach, software engineers of a new satellite project are able to overwrite or extend certain modules of the system, using the provided API.

A number of software design patterns, such as Chain of Responsibility, Strategy, Mediator and View-Controller-Model [1] patterns, have been also applied to achieve a well designed, flexible and reusable software architecture, which can not be described in detail in this paper.

It must be mentioned at this point that the provided API contains various standard solutions for all defined components of the system, which helps many satellite developers to use the system as is, without any software development.

5. APPLIED SOFTWARE TECHNOLOGY

The above mentioned software design criteria and the more generic requirement of the platform independency of the entire system has led to the choice of J2EE technology to be used for the implementation of the described software design. Figure 2 shows the typical distributed architecture of such a system, based on J2EE technology [2].
6. TELEMETRY CLIENTS

A view of the web client to the reference implementation of the system, which has been developed for the DLR satellite BIRD is shown in Figure 3. It allows the operators to access the telemetry data from all over the world via an internet web browser almost in real time during the satellite passes. It also provides tools for offline data processing of previous satellite passes, as well as direct access to the telemetry database.

New satellite projects can use the provided API to develop various clients, if the included clients do not cover their project specific requirements.

7. CONCLUSION

Using the provided development framework, micro satellite engineers are able to develop new EGSE software with respect to their own project requirements, significantly easier and faster.
Beside its flexibility, modularity and extensibility, the high performance of the system has been demonstrated by its reference implementation for the DLR satellite BIRD. A second implementation is planned for the ESA 2$^{nd}$ Young Engineers Satellite YES2. The key features of the system can be listed as

- Well designed and high performance software architecture
- No need for software development for CCSDS compliant satellite projects.
- Ease of development for extension and adaptation purposes
- Providing a number of web-based clients to access the TM data during a satellite pass through internet with reach user interfaces and different data visualizations as well as offline data processing and archiving functionalities.
- Direct access to the TM database. Any changes to the database take affect immediately, even during the satellite passes. This feature eases the procedure of the satellite testing and data calibration essentially.

8. REFERENCES

   Core J2EE Patterns, Depak Alur, John Crupi, Dan Malks