An MDA Approach for a Multi-Layered Satellite On-Board Software Architecture

Walter A. Dos Santos  
National Space Research Institute, INPE  
Space Systems Division  
walter@dss.inpe.br

Adilson M. da Cunha  
Brazilian Aeronautical Institute of Technology, ITA  
Department of Computer Science  
cunha@ita.br

Abstract

This paper proposes a Model-Driven Architecture (MDA) approach for the development of real time On-Board Software (OBS) for satellites starting from its UML modeling. The MDA definition is considered in a broad sense in which code is (semi-) automatically generated from more abstract models. A Model-Driven Development (MDD) tool tailored for real-time applications is used for software construction, which provides more general resources than just supporting UML documentation and high-level design “drafting”. This allows for model execution where one may find problems and issues that whiteboarding and document reviews are unable to find. Moreover, an incremental development, based in the 4+1 architectural views, can be effectively attained even at high-level architectural models. A layered model is suggested for the logical architectural view of the OBS domain.

1. Introduction

Software architecture has been recognized as a crucial conceptual basis for the development of non-trivial software in a myriad of application areas and by organizations of all proportions. Special attention is taken on MDA, as it is being developed to include a broad range of concepts in order to apply to all manner of software development projects. Since software systems have the ability and flexibility to deal with highly complex functions this makes them an essential part of increasing demand on artifacts for space programs. Software Engineering practices have been applied in all levels of space technology from on board systems functions to firmware.

Spacecraft computer systems and their software enhance tremendously the on board capability, but drive system costs and complexity [1].

Additionally, high quality software development of critical systems is a difficult issue. Many critical systems are developed, deployed, and used but not satisfying their criticality requirements, sometimes leading to failures.

Due to the dynamics of space software projects, a certain degree of adaptability is required to minimize costs of incorporating last-minute software specification changes in the development process. This OBS adaptability may be mapped into code with the help of an MDA approach underpinned by the use of MDD tool supporting visual modeling techniques.

In this work, by using real time UML extensions supported by a MDD tool, one can effectively deal with the high complexities of spacecraft OBS and successfully communicate and increment the software architecture as it matures.

2. MDA and Satellite OBS Architecture

Usually, various stakeholders get involved, all with specific functional and quality (QoS) requirements. The key to a successful development of software architecture is to balance the required systems functionalities with their QoS attributes. In space projects due to the plethora of issues, it is quite important to create descriptions and models from
different viewpoints so that the various stakeholders get a perspective on the system they are engaged in and they can understand. One issue that may come up, for example, is that a program manager may not be interested in technical UML models. Hence, this work suggests the adoption of the Rational Unified Process (RUP) as it identifies a standard set of views called the 4+1 Architecture Views.

The OBS, as a class of Real-time embedded systems, requires close coupling among application software, the Real-Time Operating System (RTOS), and the hosting hardware. This is against the goals of most good system architectures. To tackle this, a model-driven development tool tailored for real-time applications, Rational Rose RealTime® [2], is used as it supports UML-RT constructs for architecture documenting and provides a suitable software development environment.

The primary advantage of MDA is a unified approach to the design and development of platform-independent systems that can be easily ported from one environment to another and can be easily hosted on heterogeneous environments.

Using separation of functionality and behavior from implementation detail provided by RUP, one can reuse high-level Platform Independent Models (PIM) to generate applications that may be easily ported from one environment to other Platform Specific Models (PSM). A proposal of UML OBS Modeling is shown in [3] where it is reported a reuse feature for the case of satellites, which have a propulsion subsystem whose PIMs may be reused for simpler satellites where this functionality is absent.

### 3. Layered Logical Architectural View

In order to reduce complexity of the whole OBS, it will be adopted the abstraction of UML packages as they provide means of decomposing a potentially unmanageable object space into smaller, more-manageable pieces.

Figure 1 identifies the main packages showing the core domains for the OBS and their inter-dependencies. An expressive manner of displaying packages dependencies is achieved by a layered model, which partitions software systems into four layers (physical resources, resources access services, system services, and application).

![Multi-layered OBS Architecture](image)

**Figure 1. Multi-layered OBS Architecture**

### 4. Conclusions

This work presented some software architecture issues for the domain of satellite OBS as it encapsulates the whole design and there is a challenge on letting OBS stakeholders familiarize with them. Doing so, they would care also about architectural documentation and how they can use it to make their life easier, increase productivity, and decrease overall system development and maintenance costs.

Close attention was given to the MDA/MDD. By using real time UML extensions supported by a MDD tool, one can effectively prototype and tackle the high complexities of satellite OBS. This contributes to successfully communicating and incrementing the software architecture. Using MDA may enable design reusability and cost reduction.

Embedded developers and their tools must focus on their specific needs in order to maximize the benefits of MDA while meeting their special needs on QoS and functional attributes.

### 5. References

