Executive Summary

Authors:

Andrey Sadovykh (SOFTEAM)

Contributors:

Tom Ritter, Andreas Hoffmann, Jürgen Großmann (FHG), Alexander Vankov, Oleg Estekhin (GTI6)

Visas

Surname - Name	
Function	
Version	
Date	
Visa	

Change Log

Rev.	Date:	Author:	Description:
i1r0	20/11/2008	Andrey Sadovykh (SOFTEAM)	Initial version
i1r1	20/11/2008	Alexander Vankov (GTI6)	Editorial corrections

Need For Round Trip Engineering For Space Systems

Some year ago, the Object Management Group (OMG) introduced the *Model-Driven Architecture* (MDA) initiative as an approach to system-specification and interoperability based on the use of formal models. In MDA, *platform-independent models* (PIMs) are initially expressed in a platform-independent modelling language, such as UML. The platform-independent model is subsequently translated to a *platform-specific model* (PSM) by mapping the PIM to some implementation language or platform (e.g., Java) using formal rules.

At the core of the MDA concept are a number of important OMG standards: The Unified Modelling Language (UML), Meta Object Facility (MOF), XML Metadata Interchange (XMI), and the Common Warehouse Metamodel (CWM). These standards define the core infrastructure of the MDA, and have greatly contributed to the current state-of-the-art of systems modelling.

As an OMG process, the MDA represents a major evolutionary step in the interoperability standards. For a very long time, interoperability was based largely on CORBA standards and services. Heterogeneous software systems interoperate at the level of standard component interfaces. The MDA process, on the other hand, places formal system models at the core of the interoperability problem. What is the most significant about this approach is independence of the system specification from the implementation technology or platform. The system definition exists independently of any implementation model and has formal mappings to many possible platform infrastructures (e.g., Java, XML, and SOAP).

The UML is an evolutionary general-purpose, broadly applicable, tool-supported, and industry-standardized modelling language or collection of modelling techniques for specifying, visualizing, constructing, and documenting the artifacts of a system-intensive process. Within the system development lifecycle process, the UML readily enables communication and facilitates specifying, visualizing, understanding, and documenting problems; capturing, communicating, and leveraging knowledge in solving problems; and specifying, visualizing, constructing, and documenting solutions. The UML is broadly applicable to different types of systems (software and non-software), domains (business versus software), and approaches and processes. The UML enables and promotes (but does not require nor mandate) a use-case-driven, architecture-centric, iterative and incremental and parallel, and risk-confronting process that is object-oriented and component-based. The UML emerged from the unification that occurred in the 1990s within the information systems and technology industry.

Technical and economical advantages being promised by MDA are of a special concern for the space domain where mission life-cycle of large system often reaches 15-20 years (e.g., ISS, ATV, Columbus and many other programmes). Typically, COTS hardware and software platforms evolve tremendously throughout this lifecycle. Therefore, implementing effective and efficient modifications of legacy application software and porting it to new platforms would deliver significant benefits to space programmes in terms of cost reduction, increase of performance and flexibility.

MDA proposes to capitalise models better than systems themselves. The models represent business logic that changes slowly (especially in the space domain) and is independent from computation platform, while porting to a specific platform should be achieved semi-automatically and supported by CASE tools.

The round trip engineering is usually referred to as a process of model generation from source code (reverse engineering) followed by generation of source code from models. In this way, existing source code is converted into a model, is subjected to software engineering methods and then is converted back. Usually, this process is fulfilled on a platform specific level, while in the current activity the round trip engineering is considered on a larger scale: a model is abstracted to a platform independent level and then a new platform specific model is derived. The goal of this larger process is to extract a PIM containing the most persistent part of the legacy system and then to proceed to the deployment to a new platform through the all MDA phases. A special emphasis is put to the need of model-based verification and validation using generated tests.

The major goal of the study was (1) to develop a formal methodology for the above-mentioned process, (2) to survey the current state of the art in applicable of-the-shelf technologies and tools and (3) to demonstrate a feasibility of the approach.

Case Study and Requirements

In order to demonstrate the applicability of the previously-mentioned methodology to ground systems, ESA provided an application or a "target system", which is representative and simple enough to be able to concentrate on the methodology. This target system is a distributed file archive (FARC) enabled with a version control system.

This application had to be formally analysed using metrics in order to identify re-factoring goals. Then a platform independent model (PIM) had to be extracted, allowing capitalising of the existing know-how. In addition, PIM on conceptual level is a good means for architecture understanding and further architecture re-structuring. Moreover, a platform specific model (PSM) had to be derived for two platforms: C++ - baseline and Java, the latter represents a new migration platform. Finally, the code had to be generated for these platforms. All over the development process an early validation was required to ensure the model consistency, while on the last stage the newly re-factored system had to be validated using generated tests. Finally, the metrics analysis process had to be repeated, in order to evaluate the modernisation added-value.

Concerning the refactoring goals, ESOC emphasized the need for a GUI on Java. The modernized system is depicted in Figure 1.

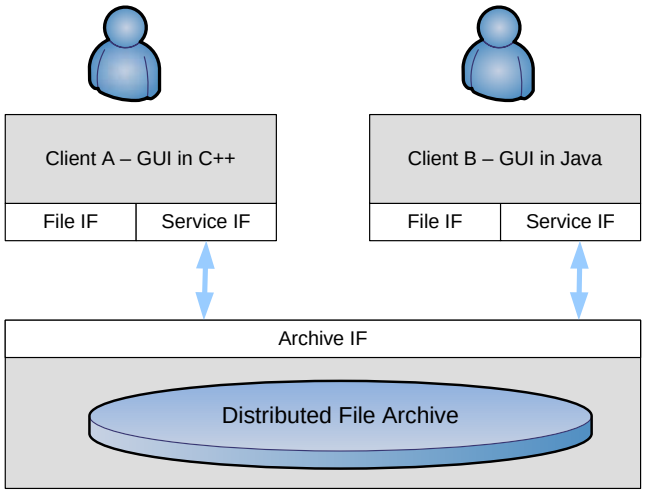

Figure 1: Logical View of Refactored FARC

Based on the SoW and the requirements definitions the following important concerns were identified for the case study.

A list of COTS frequently used within the MDA / ADM engineering and relevant for space systems was identified. Metrics-based code analysis, reverse engineering, re-architecturing, code generation and V&V techniques and tools on different abstraction layers (code, PSM, PIM) were assessed using the experience of SOFTEAM and FHG FOKUS in the MDA domain. For some phases where there was a lack of tool support (reverse engineering, PIM extraction, dedicated transformation, code generation), special actions were taken to review on-going research activities, prototype and evaluate them.

Furthermore, a significant effort was put in verification and validation rules. Specific emphasis was put on early, model-based verification and validation. In particular, FHG FOKUS's expertise in TTCN-3 and U2TP was largely employed.

The methodology and demonstrator tools chain was relying as far as possible on standards (OMG UML2, MOF, U2TP, XMI, KDM, and SPEM) and mature products to avoid proprietary solutions and ensure the capability to evolve with time.

Methodology

According to ESA requirements for the methodology, we developed an MDA / ADM methodology for the given case study.

Figure 2 depicts a flow-chart diagram for the established methodology. Initial source code was subjected to a metrics based analysis in order to calculate re-usability and maintainability metrics and to define goals of further refactoring. Then the code was reversed and a Platform Specific Model (PSM) was derived. This PSM was also subjected to the metrics based analysis.

The PSM was subjected to an abstraction transformation that aims at eliminating platform dependencies and extracting core business logic. Thus, a Platform Independent Model (PIM) was derived.

Taking into account the refactoring objectives resulted from the analysis process, the PIM was restructured in order to fulfil those goals. From this point, the verification and validation process started. PIM was used to creation of Platform Independent Test Model (PIT).

After the verification, a new PSM was generated for a Java platform. This PSM was subjected to a new round of verification. In addition the PSM was be used to generate tests for further validation – Platform Specific Test Model (PST) was created. From this point, the analysis metrics was recalculated for early evaluation of the successfulness of the re-factoring.

In the next step, the code for a given platform was generated. It was validated using the generated tests. Finally, the code was sent to the analysis process, in order to determine the added value of the whole process.

In this way, the methodology covered all the abstraction levels: Code, Platform Specific and Platform Independent levels.

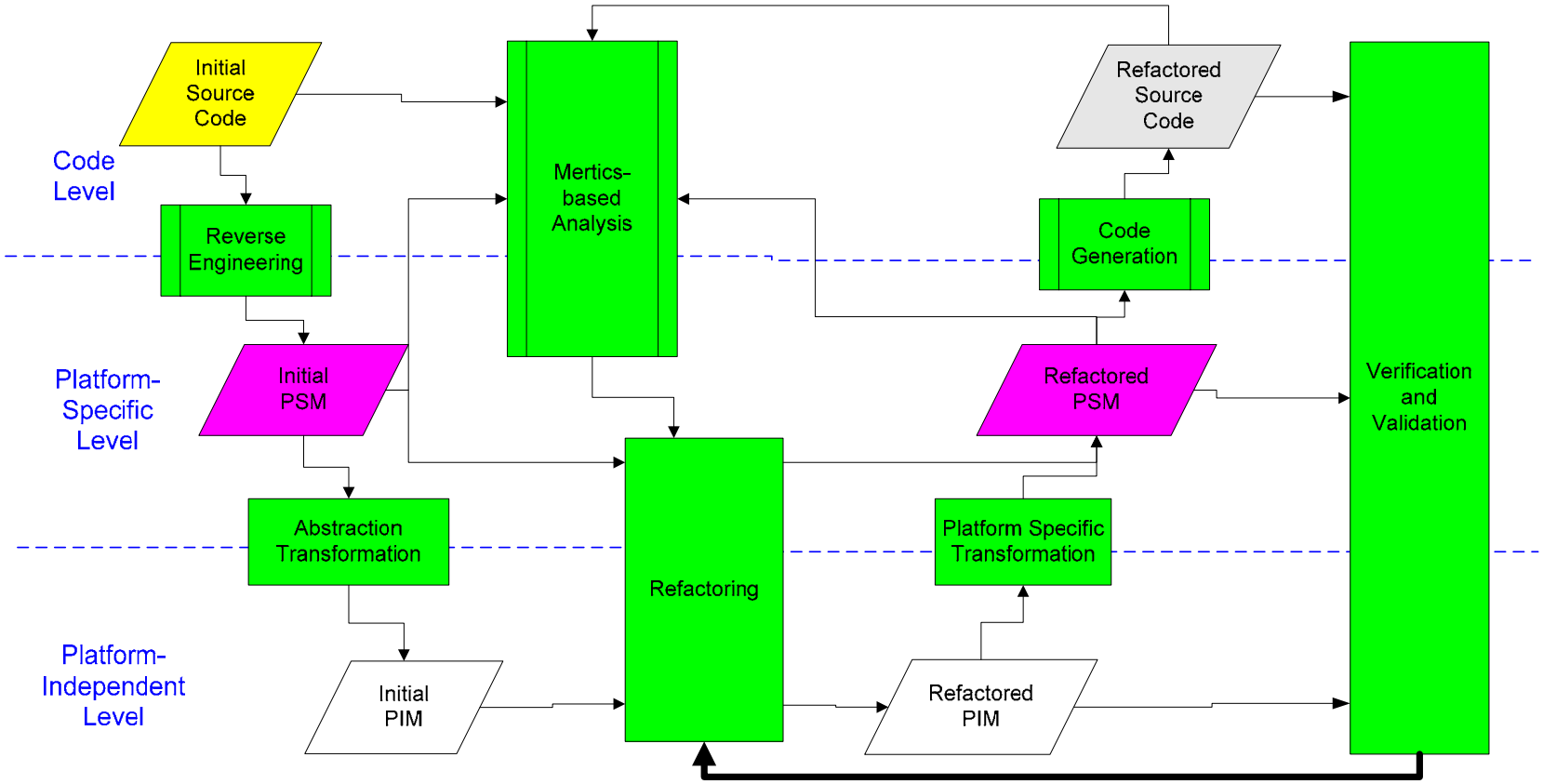

Figure 2: High Level View on Methodology

Tools

According to the requirements analysis and methodology definition the candidate tools were analysed and justification of the choice was provided.

UML2 CASE Tool had to address the following main requirements:

PIM Extraction:

- C++ reverse functionality;
- Transformation support;
- Support for customisation with UML2 Profiles.

PSM Derivation:

- Support of transformations and UML2 Profiles;
- Support of Java platform;
- Java skeletons generation and generation customisation.

Code Generation:

- Seamless synchronisation between code and UML2 model;
- Integration with code IDE;
- Configuration Management.

According to these parameters the choice of Objecteering UML2 CASE Tool was justified. This tool was used for PIM Extraction, PSM Derivation and Code Generation. Specific functionalities were developed to help in this study: PIM extraction transformation, Find&Replace for Platform Replacement, Documentation generation and etc. In addition, the requirements for the FARC system were integrated with the model including traceability establishing and documentation generation.

For the model-based testing the analysis identified the need for support of TTCN-3 and UML2 Test Profile (U2TP), the choice of the TT workbench was justified during tool assessment. For the given study, special adapters for CORBA and XML communication were developed as well as specific transformation for UML2 sequence charts for U2TP.

Results and Conclusions

The methodology established for the case study was successfully applied to the FARC system.

The Functional (PIM) model was extracted using specially developed tooling. The same process allowed for identifying the platform dependencies. This information was used in the next phase for identifying correspondent substitute libraries in Java.

During the PSM derivation, the new Java platform dependencies were integrated with the PIM model by means of specially developed tooling features. The obtained Java PSM model was used for generation of the Java code skeletons.

The generated Java skeletons were filled manually with the required functionalities applying the established migration rules. Despite usage of the MDA, the development effort was rather significant. However, the process resulted not only in the code but also in a complete UML2 model containing the documentation and requirements link. This links were automatically maintained in the model during the development process by means of the model synchronisation feature.

Metrication methodology established at the beginning of the project was successfully applied throughout all the stages: from the analysis of the original C++ application through the finally re-engineered JAVA application.

A collection of tools used for this purpose (with two key ones being available free of charge) proved its effectiveness in processing a common set of metrics for different programming languages (C++ and Java for this particular case).

However, the metrication process requires quite a considerable “human intervention” and careful analysis. It is not a straightforward procedure that could be easily automated.

The reengineering approach used in the RTE Space project resulted in the Java application that is very similar to the original C++ application in terms of metrics. The resulting JAVA code demonstrated less volume than the original C++ application, but slightly increased average complexity.

Finally, the test campaign validated the correctness of the migration from the network protocol – behavioural point of view. The model-based testing technologies applied helped in identifying several bugs in Java implementation, which were successfully fixed.

The major goals of the study were achieved: (1) a formal methodology for round-trip engineering of space systems was developed; (2) the current state of the art in applicable of-the-shelf technologies and tools was analysed; and (3) the feasibility of the approach was demonstrated.