Direction of INTRASOFT International to Software Development

Dr. Christos Georgousopoulos, Dr. Antonis Ramfos

christos.georgousopoulos, antonis.ramfos{@intrasoft-intl.com}

Abstract

This paper presents the obstacles in software development within INTRASOFT Intl. based on traditional approaches of software engineering followed, and presents the company's intention on moving towards to more revolutionary approaches, such as MDE. The advantages of adapting to MDE are also identified.

Keywords: Software development, MDE, MDD

1. Introduction

Traditional Software Engineering (SE) approaches are often driven by low-level design and coding. A typical approach consists of a number of different phases including the conceptual and requirements gathering, analysis and functional description, design, coding, testing and deployment. Though, irrespectively of the approach followed (e.g. incremental and iterative or traditional waterfall approach), documentation and diagrams produced during the first three phases rapidly lose their value as soon as the coding initiates. The gap between the documentation and code is further increased when a system is changed over time, on where amendments take place only on code level since the maintenance of documentations during development costs time and slows down the process.

An alternative to code-centric development is the model-centric approach followed by the Model Driven Engineering (MDE) paradigm. MDE refers to the systematic use of models as primary engineering artifacts throughout the engineering lifecycle, and therefore preserves the investment on requirements descriptions, design and analysis (which are expressed as models).

This paper presents the INTRASOFT Intl. direction towards the adoption of MDE, with the insight of further optimizing the software development production line to support customer solutions with a maximum level of reliability, scalability, cost-effective and on-time products and services. The rest of the paper is organized as follows: section 2 provides a brief description on INTRASOFT Intl. and the software development cycle within the company, section 3 discusses MDE with emphasis to

MDD by identifying the advantages of utilizing this approach. The paper concludes with INTRASOFT Intl.'s intention on practicing MDE.

2. INTRASOFT International S.A.

INTRASOFT International S.A.[INTRASOFT(2007)] is a leading European company in the area of Information and Communication Technology (ICT) services. It has a broad portfolio of business activities addressing a wide range of customers mainly in the public sector, including international and national organizations. Its extensive service exposure offers innovative and added-value solutions of high quality with measurable results for the company's clients. In particular, the company's offering is grouped along 4 service lines: Application Development & Systems Integration, Networking & Security Services, e-Government Solutions & Managed Services, and Outsourcing, Consulting and Professional Services.

In addition, INTRASOFT Intl. realizes that research in ICT business is one of the critical factors in a company's growth, as it transforms knowledge and ideas into services and products. INTRASOFT Intl. has participated in several research projects in the FP5 and FP6 Programme. The focus of the company's research activities is at e-government and e-business areas, information system management, knowledge management, interoperability and integration technologies, security and risk management, and software engineering.

As a subsidiary of the INTRACOM Group, INTRASOFT Intl. was established in October 1996 to better serve Western Europe. Since then INTRASOFT Intl. has achieved impressive organic growth. Consolidated organic revenue in 2006 reached 50 million. At the end of 2006, INTRASOFT Intl. employed over 730 professionals working in Belgium, Luxembourg, Athens and Romania, representing 20 different nationalities and mastering more than 15 languages.

2.1 Software development within the company

INTRASOFT Intl. follows a global offshore development strategy of customer solutions using traditional code-centric Software Engineering methodologies Based on such approaches, the code is the driving force of software development and the only phases in the development process that are really productive are coding and testing. The documents and corresponding diagrams created in the first three phases (requirements gathering, analysis, design) rapidly lose their value as soon as the coding starts, where the connection between the diagrams and the code fades away as the coding phase progresses. In this instance, during software development phases and after the delivery of a complete solution - in the aspect of maintenance, traditional SE methodologies lack the ability of providing support on:

- Portability and reusability: Software industry is characterized of its ever-increasing rate of new technologies that emerge. Developers need to follow these new technologies since (a) they provide solutions to real problems, such as XML[XML (2007)] for interchange, J2EE[J2EE (2007)] and .NET[.net (2007)] for platform independence, and (b) they are demanded by the customers e.g. web-interfaces. The porting of a legacy system to new technologies requires extensive code-rewrites and testing. As a consequence, investment in previous technologies loose value and they may even become worthless.

- Maintainability: keeping the application architecture and application code consistent is a very difficult procedure on complex large-scale systems. Developers are mostly concentrate on code amendments rather than on updating appropriate architectural documentation as well. Therefore, the architectural documentation that initially forms the high-level specification of the code to be produced, as code propagates and evolves, it tends to loose its significance. Maintenance involves the customization, optimization and correction of implementation problems, where it prerequisites the identification of the part of the system than needs to be amended. The importance of reliable documentation lies on the fact that developers may better comprehend a system by focusing on its formal documentation (such as UML diagrams) that express the architectural structure, rather than on exploring code-segments consisted of thousand of lines.

- Quality of code: the quality of code produced by a developer is directly related to his expertise or his ability on choosing well-accepted programming patters on resolving specific issues. The development of a system involves the collaboration of a number of software developers of different kind of skill levels. Therefore, the quality of code cannot be guaranteed, as it is possible with model-centric approaches on where code is automatically generated based on predefined rules.

As part of INTRASOFT Intl.'s commitment to always search for new ideas, higher performance and service improvements for its clients, it is believed that innovation can foster the use of new technologies, methodologies and culture that can have a long-lasting beneficial effect not only for the customers but also for the company's business core. Based on the skills in understanding the challenges and market complexities, with the objective of constantly satisfying its customers needs, INTRASOFT Intl. envisaged that moving towards to more revolutionary approaches in Software Engineering methodologies will benefitits software development production line by providing solutions to the above mentioned problems. In this direction, INTRASOFT Intl. is currently investing on the exploitation of Model Driven Engineering.

3. Model Driven Engineering and MDD

MDE is an approach to the full lifecycle integration and interoperability of enterprise systems comprised of software, hardware, humans and business practices, providing a systematic framework to understand, design, operate and evolve all aspects of such systems, using engineering technologies and tools. MDE is put forward and structured by the Object Management Group (OMG) initiative called Model Driven Development (MDD)[OMG (2007)].

Two popular variants of MDD are Model Driven Architecture (MDA) and Model Integrated Computing (MIC). The main differentiation between the two standards lies on the type of formal language used to represent system elements and their relationships, as well as their transformations to platform specific artifacts. MDA uses OMG's general-purpose UML[UML (2007)], where MIC uses domain-specific modeling languages (DSMLs).

MDD is a part of a broad effort across the industry to raise the level of abstraction in how to develop systems. MDD is about using modeling languages that make possible to program systems at a higher level of abstraction, than it is possible using programming languages. The MDD methodology is based on the derivation of three models and appropriate transformation rules. A representation of the MDD artifacts and their relationships is illustrated in Figure 1.

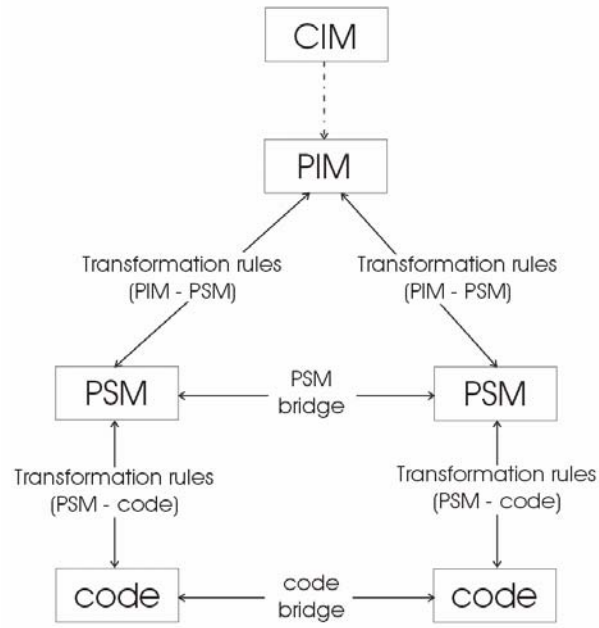

Figure 1. *MDD workflow*

The first model to be defined is the Computational Independent Model (CIM[1]) that captures the system requirements of a particular business domain and identifies the key concepts or business entities along with their relationships. Then the system is modeled in a formal but completely technology independent representation within a Platform Independent Model (PIM). In continues, a set of appropriate transformation rules are necessary to be defined in order to transform a PIM to a Platform Specific Model (PSM). A PSM is a view of the system from the platform specific viewpoint. Typically, a PSM combines the specifications in the PIM with the details that specify how the system uses a particular type of platform. The final stage involves the definition of transformation rules to transform the PSM to actual code.

3.1 Advantages and disadvantages of MDD

In contrast to code-centric approaches, MDD shifts the focus of software development from writing code to modeling. Apart from providing a solution to the problems identified in section 2.1, also:

- Improves productivity: the effort required in the MDD approach on developing a system is mainly based on the modeling of the PIM and the transformation rules. Once the rules for the transformation of PIM-to-PSM and PSM-to-code are defined they can be re-used and be applied to the development of other systems of relevant platform interest (increase return on technology investments). In addition, the advantage of developers on shifting their focus from the code to PIM is twofold. Firstly, developers' work is minimized, as platform-specific details need not be designed and written down since they are already addressed in the transformation rules. Secondly, paying more attention to solving the business problem at hand, rather than concentrating on code writing, results in a system that fits much better with the needs of the end users.

- Increases portability across middleware vendors: The PIM separates the specification of functionality of a system from the specification of implementation of that functionality on specific platform technology. In this instance, when there is a need to switch between middleware platforms (e.g. from J2EE to .NET) effort is only required on the definition of a new PSM, preserving in that way the initial architecture of the system intact. The existence of a great collection of open-source MDD tools and all-ready-made customizable transformation rules assist the transformation of a system to adhere on different middleware vendors, as well as to new emerging technologies.

- Fosters interoperability: MDD apart from supporting the generation of multiple PSMs from a single PIM, it also enables the definition of PSM-to-PSM relationships, referred to as bridges. Bridges enable the transformation of concepts from one

[1] Usually referred to as a domain or a business model

platform into concepts used in another platform enabling the developing of distributed systems that are made up of components running on different platforms/tiers.

- Optimizes maintainability, code consistency, quality & quantity: the PIM is a high-level representation of the system. Changes made to the system will eventually be made by changing the PIM and regenerating the PSM and the code. In practice today, many of the changes are made to the PSM and code is regenerated from there. Good tools, however, will be able to maintain the relationship between PIM and PSM, even when changes to the PSM are made. Changes in the PSM will thus be reflected in the PIM, and high-level documentation will remain consistent with the actual code. Consequently, investment on models (i.e. architectural documentation) is not lost in contrast with the code-centric approach. In addition, the quality of code produced by models is increased due to the fact that well-accepted design patters may be incorporated into transformation rules providing developers (irrespectively of their skill level) the ability to use the same underlying design patterns, since the code is generated in the same way each time. According to OMG, today's tools typically automate 50% to 70% of the PIM-to-PSM transformation, where the automation of the PSM-to-code transformation is typically 100% or nearly so[OMG (2007)].

- Enables testing and simulation at an early stage of development: since the developed models can be used to generate code, they can equally be validated against requirements, tested against various infrastructures and can be used to directly simulate the behavior of the system being designed.

The disadvantage of MDD lies within the scope of configuration management systems. This is due to the limited functionality provided by CVS systems[CVS (2007)] on managing diagram files (models). Although, basic operations such as "file locking" and "versioning" are currently supported, more complex functionalities like "brunch and merge" are not. Of course, such capabilities are expected to be supported in the near future since most of MDD models are now expressed in OMG-XMI [XMI (2007)] format (XML representation).

To conclude, MDD provides the means to specify systems at a higher level of abstraction. As a result there is an increase on the longevity of a system, since the business logic and architectural specifications are not tied-up with a specific technology - investment in business model is preserved even as technologies come and go. Moreover, by raising the level of abstraction quality is improved, because emphasis is given on determining the business requirements for an application rather than on the realization of the application using a particular technology standard.

4. Conclusion

INTRASOFT Intl. is investing on an IST European project called ELLECTRA-WeB to exploit Model Driven Engineering. The overall aim of ELLECTRA-WeB is to specify, develop and test an Open Source Application Framework for electronic Public Procurement and a set of guidelines that will enable, on the one hand, Western Balkan ICT developers to cost-effectively develop, and, on the other hand, Western Balkan Public Administrations and Suppliers to easily adopt and use interoperable, user-friendly, secure and EU compliant electronic Public Procurement solutions.

This will be achieved by focusing on state-of-the-art software engineering technologies, namely, Model Driven Engineering and Service Oriented Architecture, in order to reduce the development cost and shortening the development time of EU compliant electronic Public Procurement solutions. The Innovation and Solutions (R&D) department of INTRASOFT Intl. has selected to practice MDE on e-Procurement domain due to the company's long-standing experience on e-Tendering/e-Procurement solutions market. Other projects in this domain include[INTRASOFT (2007)]: SIMAP, e-Procurement Pilot system (Romanian Government), e-Tendering Pilot System (Greek Government), EPSS (DG Research), SYMMETRY (DG Education and Culture), Reliable e-Public Procurement System (RePublic/eTen), and electronic Public Procurement in Europe (e-PROCSEE/eTen).

It is strongly believed that research on exploiting MDE within such a large-scale project will provide the R&D dept. with the required expertise and knowledge on defining methodologies and guidelines to support and resolve current obstacles within the company's software development department.

References

INTRASOFT International S.A., http://www.intrasoft-intl.com *(last visited 2007)*
OMG - The Object Management Group, http://www.omg.org *(last visited 2007)*
UML - Unified Modelling Language, http://www.omg.org/UML *(last visited 2007)*
XML Extensible Markup Language, http://www.w3.org/XML *(last visited 2007)*
.net - Microsoft .net Framework, http://www.microsoft.com/net *(last visited 2007)*
J2EE - Java 2 Enterprise Edition, http://java.sun.com *(last visited 2007)*
XMI - XML Metadata Interchange, http://www.omg.org/technology/documents /formal/xmi.htm *(last visited 2007)*
CVS - Concurrent Versions System, http://en.wikipedia.org/wiki/Concurrent_ Versions _System *(last visited 2007)*