Model-Based Software Development for Interactive Systems: Ready for Practice?

Roland Petrasch[1]

***Abstract*—**The development of interactive media and software systems for practice includes different disciplines like requirements management, usability engineering, and software engineering. Analyzing, documenting and formalizing functional and ergonomic requirements so that they can be used for the design and implementation of interactive software systems, is a complex task for the domain experts, requirements analysts, and usability engineers. A critical aspect is the transformation of the requirements in the direction of software development: How can the risk of misinterpreting requirements and of delivering wrong software components be minimized? One positive factor could be the usage of model-driven or model-based techniques that contributes to lowering the risk of semantic gaps or misinterpretations. Model-based user interface design (MBUID) focus' on interactive systems: Precise and concise requirements can be transformed automatically and the generated code is ready to use or can be modified by software or usability engineers. Usability aspects are part of the models so that ergonomic requirements can be considered easily in an early project stage. This paper contributes to the discussion about the practical application of MBUID.

***Keywords*—**Model-based User Interface Design, Model-Driven Development, Agile project management, Scrum, Code Generation, MDA.

I. INTRODUCTION

For software systems *conceptual models* can be used in the context of the requirements specification. This task is usually performed with a natural or formal language, e.g. UML (Unified Modeling Language [1]). Gathering, analyzing, formalizing, verifying, tracking and documenting requirements is a complex task for the requirements engineers and system analyst. But complexity and does not stop here: Another challenge is the transformation of requirements in terms of software engineering, i.e. the design and development of software: In order to ensure that the implemented systems meets the requirements, comprehensive testing is necessary.

Relating to interactive software systems, i.e. software with a user interface (UI), formal requirements specification models describe data, interactions and navigational structures inter alia. A use case diagram is a conceptual model and a good example for a UI-relevant specification of functions because it contains information about the users (actors) interacting with the system (functional decomposition). *Model Based User Interface Design* (MBUID) leverages formalized models for the generation of code for interactive software systems [2].

Roland Petrasch[1] is with Thammasat University, Faculty of Science and Technology, Computer Science Department, Pathum Thani 12120, Thailand (corresponding author's phone: +66 926104285; e-mail: roland.petrasch@cs.tu.ac.th).

A platform independent model (PIM) as a conceptual model is on a high abstraction level and is used during the requirements specification. An abstract user interface (AUI) is part of a PIM. In contrast to PIM a PSM (platform specific model) specifies a software system for a specific platform [3]. Therefore a PSM is similar to a software design model. For the UI part these models are called CUI (concrete user interface). The last step is the usage of code generation tools in order to create a PSI (platform specific implementation) containing the FUI (final user interface). Fig 1 shows this process of model transformations.

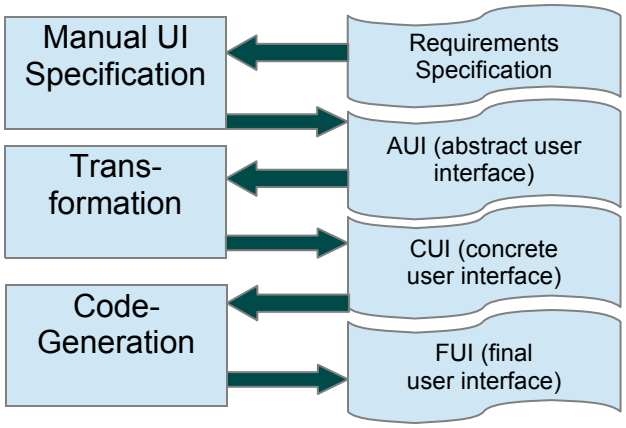

Figure 1: Model Based User Interface Design: Model Transformation and UI Code Generation

The effect of MBUID can be an increasing productivity and a better software quality when applied properly. But there are also challenges and tradeoffs when it comes to an application of this model-driven approach in the context of real-life projects. The following chapter takes a closer look at this.

II. PRACTICAL ASPECTS OF MBUID

In projects for the development of interactive software project management, quality management, and software engineering are important success factors:

1) Project Management

Agile techniques for PM (project management) are widespread and have proven successful, e.g. Scrum-based approaches [4] [5]. For MBUID it is possible to use Scrum, but it needs customization and extensions.

One most important aspect here is the setup of two separate projects: One that is responsible for the model-based development infrastructure and the other for the customer / application

development project. The infrastructure (or MBUID development environment) project delivers the meta-model(s) and tools (transformers and generators) that can be used by the customer project that specify, generate and implements the application software for a concrete customer. Fig 2 shows the connection between these two projects.

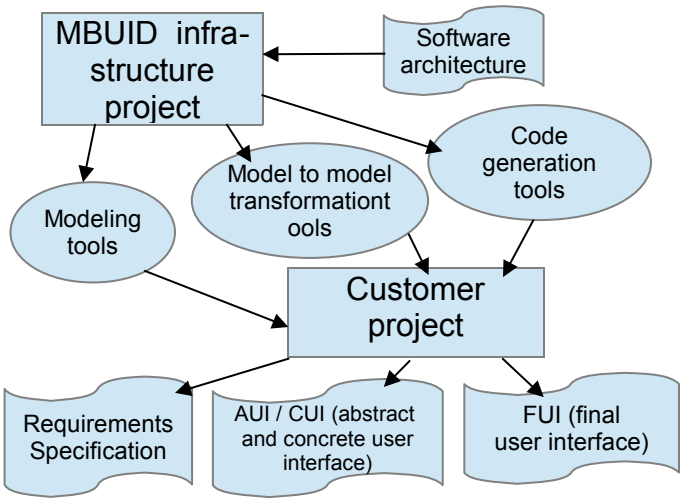

Figure 2: MBUID infrastructure project and customer project

Infrastructure and customer projects are two independent projects, but they have connection points: The customer projects communicates the requirements for a specific technology to the infrastructure project that develops the MBUID tools needed for the customer project.

When a Scrum-based approach is used, the customer project has a product owner who is able to formalize the requirements and create the models necessary for the MBUID tools. The team uses the model transformer and code generators. Also manual programming is performed by the team. The infrastructure project has a product owner who is a software architect for the technologies that are used as the target platforms in the customer projects. The team creates the modeling tools, model transformers and code generators.

2) Quality Management

Constructive and analytical quality assurance measures are to be applied in MBUID projects similar to normal software projects, but the focus here is on model quality. Therefore, specialized QA techniques are needed, e.g. modeling guidelines, DSL user manuals, model validation and verification methods and tools.

Not only manually created code, but also transformation tools and code generators need testing. On the system level, system and acceptance testing is necessary: requirements coverage is a very important aspect, because the customer project must make sure that all functional and non-functional requirements are fulfilled.

3) Software Engineering

The development of MBUID tools has some similarities to normal application programming with a object-oriented programming language like Java. Languages like MTL, QVT, OCL, MOF used for model-driven development are object-oriented, but there are also fundamental differences: The creation of meta-models or DSLs (domain specific languages) and the programming with a model-to-model (M2M) or model-to-code (M2C) transformation language requires specialized software architects and engineers.

Transformation tools for instance are developed on the level of meta-model elements that must be transformed or mapped: source meta model elements are transformed into elements of the target meta-model. When the transformation is applied to an model that conforms to the source meta-model (elements are instances of source meta-model elements), the transformer creates or modifies the target model (elements are instances of target meta-model elements).

4) Other aspects

One question arises quite quickly during discussions about MBUID: Is there a tool chain available that works out of the box for my project? The answer is yes and no: Yes, there are powerful tools for modeling, DSL creation, M2M transformation and code generation on the market. For (meta-)modeling and DSLs MOF (Meta Object Facility) and UML (Unified Modeling Language) tools can be used, e.g. eclipse EMF/UML, for M2M transformation also an eclipse-based OVT-O (Query / View / Transformation - Operational) implementation is available and for M2C (code generation) Acceleo M2T (MOF MTL implementation) from Obeo (www.obeo.fr) is widespread.

But there is also a "no", because the individual MBUID infrastructure must be setup using the tools mentioned above: Specific DSLs for the customer domain and specific transformers and generators for software platforms and technologies must be developed or at least customized by the infrastructure project before the customer project(s) can use them.

As mentioned above, experts for MBUID are essential for its successful application in practice. The skills necessary for MBUID projects are software architecture, meta-modeling, DSL design and implementation, transformation and generation tools development.

Another very important aspect is the introduction of MBUID in practice. A step-by-step approach should be preferred: Instead of trying to formalize and model every part of an interactive software system in the first place, only a few parts like the controllers and entities (data model) should be covered by the MBUID tool chain at the beginning.

After successful evaluation more and more aspects can be included in the model-driven approach. Style guidelines and usability or HCI (human computer interaction) patterns can be included in MBUID making it a powerful approach for a better user experience [6].

III. Summary

The practical application of an model-based development approach for interactive software systems is possible: An agile project management for the infrastructure and the customer project is recommended. Also quality assurance measure to ensure a high level of quality is an essential part of MBUID in real-life projects. Software engineering methods and tools are available and have been proven successful. However, experts for MBUID are necessary in order to provide the model-driven development environment for the customer projects.

References

[1] Object Management Group (OMG): UML – Unified Modeling Language (UML) Superstructure Specification. Version 2.4.1. 2011

[2] Roland Petrasch: Model Based User Interface Development and HCI Patterns: The Registration-Pattern as an Example. In: Songsak Rongviriyapanich, Roland Petrasch (Eds.): Proceedings of Software Engineering and Quality Assurance. Technical Report #1, Thammasat University, Beuth Hochschule, Logos Verlag, 2012

[3] Object Management Group (OMG): MDA Guide.Version. 1.0.1, omg/2003-06-01, June 2003

[4] Ken Schwaber: Agile Project Management with Scrum. Microsoft Press, 2004

[5] Roland. Petrasch, Torben Franzke, Songsk Rongviriyapanich: Einführung von Scrum in einem Software-Entwicklungsprojekt der ContiTech AG. In: Hanser et al. (eds): Vorgehensmodelle 2013: Vorgehensmodelle – Anspruch und Wirklichkeit (WIVM2013). 9.-10. Okt. 2013, Lecture Notes in Informatics, Gesellschaft für Informatik, Bonn, 2013, p. 236

[6] Roland Petrasch: Model Based User Interface Development with HCI Patterns: Variatio Delectat. PEICS'10, June 20, 2010, Berlin, Germany, ACM, 2010