Model Driven Development Challenges in the Automation Domain

Detlef Streitferdt*, Georg Wendt*, Philipp Nenninger*,
Alexander Nyßen†, Horst Lichter†
*ABB Corporate Research Center Germany
Wallstadter Straße 59, 68526 Ladenburg
Email: {detlef.streitferdt | georg.wendt | philipp.nenninger}@de.abb.com
†Research Group Software Construction, RWTH Aachen University
Ahornstraße 55, 52072 Aachen
Email: {any | lichter}@swc.rwth-aachen.de

***Abstract*—Model driven development has evolved to a mature methodology and technology usable for some industrial settings. Within the automation domain it is an upcoming approach. This paper addresses challenges present in the automation domain when it comes to the usage of model driven development. Quality, life cycle, legacy systems, mental approach and safety challenges are briefly discussed.**

I. Introduction

Model driven development (MDD) has been around for several years and proposes the usage of "models at different levels of abstraction and performs transformations between them in order to derive a concrete application implementation" [1]. One way to implement MDD is using the Object Management Groups (OMG) Model Driven Architecture (MDA). Established in 2001 MDA is a base architecture for OMG standards. The terms used in the field of modeling differ a bit why the OrVia research project elaborated a comparison of "model based" versus "model driven", see [2].

- *based* - A development process is used and described with models which are further developed. Restricted to one platform.
- *driven* - A development process is used and shall be largely automated. Models are transformed one to another and shall be reused amongst different platforms.

Thus, the main focus of MDD is on models which are used as central concepts and are the basis for an automated system development process. Although a system consists of hard- as well as software components, in the automation domain the hardware mainly imposes constraints on the system that have to be addressed by its software. In [1] a model is defined as "a coherent set of formal elements built for some purpose that is amenable to a particular form of analysis. A model is expressed in a modeling language at some abstraction level which in itself can be defined by meta-models". With this broad definition in the end *everything* that can be processed with a computer can be used to represent a model. There is a distinction between the "mental model" as an abstraction of a complex entity (which may be again a model) and a representation of a model, which is used to share the mental model between stakeholders (e.g. developers, project managers, customers). By examples this definition can be further explained. An Excel sheet or a Word file can be a representation of a model, any UML diagram can be a representation of a model, MATLAB/Simulink is working with models and its representations, source code can be a representation of a model and even pieces of ASCII text can represent models. The development work in an industrial environment creates models of all kinds that will be used in the MDD.

In section II current trends and developments in the domain of MDD are explained to build the basis for the challenges formulated in the following section. In section IV the automation domain is briefly introduced and challenges in terms of MDD are summarized.

II. Model-driven Development-State-of-the-Art

To successfully apply MDD, models have to be available. Thus, means to express domain specific structural and behavioural aspects have to be selected. Transformations from one to another model have to be defined and implemented. This could be a rather effort consuming task given the current tool landscape. Several tools have transformation capabilities and for complete tool chains there are ideas like the ModelBus [3] to unify a tool landscape with different databases using a service infrastructure, or GeneralStore [4] offering a central meta model with XMI import / export capabilities for the integration of modeling tools.

Any changes in the selected tool chain affect the transformations between the tools and thus the models which causes adaptation efforts for the transformations. This directly requires to establish an environment for hosting models with their transformations, with effort to be spent for the maintenance of the environment.

MDD promises the following benefits [1]:

- Time savings by code generation. The development task takes place on a level above the source code. Only models are used, source code is always generated. In addition, architectural elements can be re-used, what results again in time saving.

Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International , pp.1372-1375, July 28 2008-Aug. 1 2008
doi: 10.1109/COMPSAC.2008.156

- Quality improvement (performance, availability, security, modifiability, scalability, reliability) by using well established and tested transformations.
- Cross-platform development and enhanced platform migration. Changing the platform (HW) by changing the generation engine is easier than re-implementing the code developed specifically for a given platform.
- Models are used to improve communication and interaction of different domain experts. As a central concept for any communication models have defined semantics and reduce common misunderstandings.

As model transformations are an important issue of MDD, it is important to know what has to be transformed and how the information is structured. The OMG established the Meta Object Facility (MOF) which makes use of meta-models on four levels:

- M0 - Concrete data, e.g. generated code.
- M1 - Models defining M0 data, e.g. any data, Unified Modeling Language (UML) or object models.
- M2 - Meta-model defining M1 models. For example the UML-meta-model defines the concept of e.g. classes and relations between classes.
- M3 - Meta-meta model defines M2 models. Here are model elements that can have some kind of connection to other model elements.

MOF in the form of Essential MOF, a reduced MOF, is used in the Eclipse Modeling Framework (EMF) [5]. EMF is used to implement the open source XML Schema Infoset Model (XSD), Service Data Objects (SDO), UML2, and Web Tools Platform (WTP) projects at Eclipse. In addition EMF is used in commercial products, such as Omondo EclipseUML and IBM Rational and WebSphere products, [5]. Here, it is important to notice that several development plug-ins are making use of the Eclipse platform with EMF and the underlying model to store data. Manipulations of any models can be done using the Xtend/Xpand language, by openArchitectureware.org (OAW). With this, arbitrary transformations are possible.

Another concept are Domain Specific Languages (DSLs). A graphical [6] or textual[1] (e.g. a programming language like C or even the Business Process Execution Language, BPEL) language is defined such that it is possible to describe and solve any problem in the domain in question. Thus, there are only two transformations, the first from the DSL to source code and the second from the source code to machine code, which is performed with a standard compilation environment.

III. Characterization of the Industrial Automation Domain

This paper focuses on the automation domain with products for factories like paper mills, oil & gas or energy production systems. These large plants are built and assembled from smaller parts referred to as embedded systems which have an own development process prior to the manufacturing of the plant. This paper discusses challenges for the MDD of such embedded systems.

Embedded systems in the automation domain are a broad application field, from very small 8-bit/1kB devices up to large 32-bit/40MB devices. Temperature, flow or pressure sensors for the process industry (e.g. chemical plants or food production), motor starters, softstarter or frequency converters are all based on such embedded systems.

IV. Challenges in the Industrial Automation Domain

Model based development is state-of-the-art in traditional software development and has been an emerging topic over the last years for embedded system development in the automation domain. Several issues are present and need to be addressed when making use of MDD concepts with regard to products, development and methodology.

- The quality of small products has a big impact on the overall system.
- The overall life cycle of a product is often over ten years.
- Most development efforts are based on existing systems (old systems or even very old systems).
- MDD requires a model based mental approach.
- Safety as being defined in IEC61508 has to be addressed in many systems.

The following paragraphs briefly present the challenges present along the lines of introducing, using and maintaining MDD within the automation domain.

The Quality Challenge. A failure of small products (the embedded systems) has a big impact on the overall systems (e.g. a plant) reliability and uptime. Downtime costs of huge factories / plants are very high and can go up to several million dollars per day. As stated in the last section, quality is a key issue of MDD. Thus, the challenges presented in [7] are valid for MDD as well and have been taken as basis for the following list of quality issues, to be addressed in MDD:

- introduction of new technologies
- uncertainty in introducing new tools or tool chains
- simulation

Based on own experiences with hard- and software technologies (e.g. new processor types, peripherals, UML profiles or development methods), changes evolve within a two to four year cycle. For MDD, only changes leading to an decreased development time or effort, are relevant. Either a technology changes towards an enhanced version that is worth looking at, or new technologies are developed, which are again worth looking at. Estimations on the impact of a changed / new technology versus staying with an existing setup are very hard and here, prediction/estimation models are needed to easily assess intended changes.

In a similar league as the technology changes the introduction of new tools or tool chain bares a high uncertainty in estimating the impact of such changes. The models of a system with their interrelations form the valuable asset for a company. Tools working on such data are just enablers for the MDD idea.

[1] OAW has proposed a textual modeling framework based on Eclipse and EMF named Xtext

Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International , pp.1372-1375, July 28 2008-Aug. 1 2008
doi: 10.1109/COMPSAC.2008.156

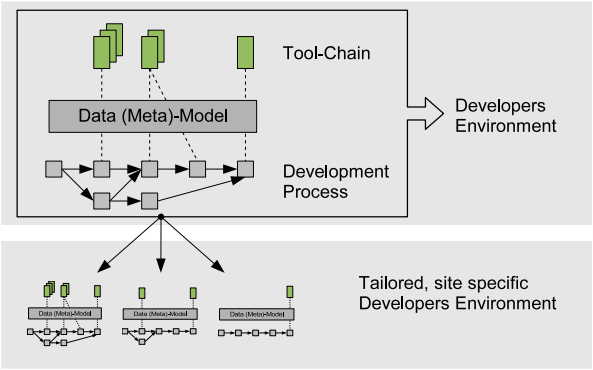

Fig. 1. Developers Environment with Tailoring

Capabilities and features of tools are of course different and lead to different maturity levels that need to be considered. Thus, a tool evaluation before its integration is a prerequisite. Even with good preparations prior to the integration of a tool into the MDD landscape of a company, being locked into a specific tool, simply because of its proprietary interfaces is a risk that has to be considered. The challenge is to foster standardization efforts towards inter tool data exchange models or working intra tool import / export interfaces. The vision, depicted in figure 1, is a combined tool method knowledge base, called the developers environment. The development process as well as the tools in the tool chain can be tailored to the specific needs of a given development department or a project. The integration of a method with all the needed tools, together with a model driven approach offers quite high saving potentials. For the marine, automotive and aerospace domain a developers environment with simulation capabilities was built, see [8]. The resulting 30% development time reduction shows the high potentials but also the needed integration of tools to reach such savings.

A well known quality aspect in systems development is the time of error correction - the earlier the better. Once models are available, tool supported simulation is possible and supports the identification of errors in early stages, at requirements level. This was done in the aforementioned project with the 30% development time reduction, see [8]. In this project, simulation was done with Hardware-In-The-Loop (HIL) and it turned out, that the simulation was a key success factor to reach such a high development time reduction. For HIL simulation as well as model based debugging special hardware and a model simulator (which is a software tool) are needed. All relevant parts of the environment have to be simulated to ensure correct simulation results. The simulation of the exact timing behaviour can also be tool supported, see ChronSim [9] for example.

THE LIFE-CYCLE CHALLENGE. A product life cycle of often more than ten years, as present in the automation domain, directly leads to the known issue of how to be able to change, maintain or only re-do any development steps from the very first product version at the end of the product life cycle? Tools, operating systems, target hardware or development hardware changes over time, a company in the automation domain has to make sure that all the models initially developed for a product are still usable even if the product life cycle comes to its end. In particular, it should be possible to execute all transformations developed at the beginning of the project to make corrections to the model or introduce enhancements.

MDD of course offers means to overcome the challenge of changing hard- and software by increasing the abstraction level inherently present in the models. The basic idea is, that the models are stable over time, and by improved transformations to new target platforms the software for an embedded system can be generated at any time. Within the automation domain there is still the challenge left to validate whether this concept holds over time. A solution idea is to have improved, model-based testing methods and tools, which enable a validation of newly generated code for new platforms.

THE LEGACY CHALLENGE. Any development method in the automation domain has to address the integration of previous development efforts. In many cases legacy hard- and software has to be used and re-used as much as possible. Inside a MDD project such re-use is only possible by describing a legacy component by its exact interfaces. Once this is done the newly created model element, a component, can be used in the current system. The drawback in this approach is that legacy systems could have been developed without the component idea in mind. Thus, even if an interface around a piece of source code can be identified, the resulting component is rather specific. This can be addressed by refactoring the software to get to better re-usable components. The challenge is to find ways to reduce the effort needed to integrate legacy systems or components into the models of MDD projects.

THE MENTAL APPROACH CHALLENGE. MDD, by construction, requires a development engineer to think in terms of models. In addition to this, the well known source code is still present and needed in the system, but this piece of information is not meant to be used or even changed by a developer. Most current tools have a round-trip-engineering support, to enable the manipulation of source code. This eases the migration to MDD. The future vision goes towards embedded system development without touching the actual source code.

Based on estimations, the average development experiences of developers in the automation domain are around ten years. These experiences are mainly in the field of traditional C-Code development. Clearly the changes in the mental approach towards MDD require training efforts, tool chain changes, and refactoring, as well as re-modeling of existing systems. With the given market timing constraints, changes in the development method as well as changes of the tool chain, can only be made gradually based on a migration plan. The challenge is to get to a high quality estimation of the efforts needed to get to the mental approach of MDD.

MDD should be an integral part of an engineering education in e.g. computer science or electrical engineering. The

Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International , pp.1372-1375, July 28 2008-Aug. 1 2008
doi: 10.1109/COMPSAC.2008.156

challenge is the integration of the structured MDD in the educational system of universities. In addition, domain specific add-ons need to be addressed within companies.

THE SAFETY CHALLENGE. An increasing number of embedded systems in the automation domain are certified or will have to be certified according to the safety standard IEC61508. MDD on the one hand enables a structured and documentable development path to a product, but on the other hand currently offers only little support in directly addressing safety issues in models or its transformations. In the avionic, railway and automotive domain safety enabled tools are available, like Esterels Scade [10]. The challenge is to elaborate more on safety certified model transformations and the integration of best practise knowledge out of non-automation domains.

V. CONCLUSION

MDD with models and transformations between models as key concepts, has evolved to a mature methodology and technology partially usable in an industrial setting. For the automation domain an MDD analysis revealed challenges that have been discussed in this paper.

To keep and enhance the product quality calculation models for better estimations are needed. Improved integration options for a tailorable tool landscape are needed to secure an investment in MDD technology and methodology. In addition, the tools shall offer safety certified model transformations. At the same time legacy system integration into MDD projects should consume a reduced effort. For the products with a long life cycle the promises for a long lasting model driven developers environment have to be validated. Finally, high quality estimations are needed for the costs the mental change towards MDD causes.

Further research in the MDD arena will have to dig deeper into the challenges towards solutions, which are applicable in the automation domain. Personalized migration paths need to be developed, starting with the current status towards model based development to finally reach the model driven development goal.

REFERENCES

[1] Aram Hovsepyan and Stefan Van Baelen and Bert Vanhooff and Wouter Joosen and Yolande Berbers, "Key research challenges for successfully applying MDD within real-time embedded software development," in *International workshop on embedded computer systems: architectures, modeling and simulation (SAMOS 2006) edition:6 location:Samos, Greece date:17-20 July 2006, Lecture notes in computer science vol:4017 pages:49-58*, 2006.

[2] OrVia, "Orchestrierung und Validierung kooperierender Systemkomponenten," Website, 04 2008. [Online]. Available: http://www.ids-scheer.com/de/orvia

[3] Xavier Blanc and Marie-Pierre Gervais and Prawee Sriplakich, "Model Bus: Towards the Interoperability of Modelling Tools," in *Lecture Notes in Computer Science*, vol. 3599/2005. Springer Berlin / Heidelberg, 2005, pp. 17–32.

[4] Clemens Reichmann and Daniel Gebauer and Klaus D. Müller-Glaser, "Model Level Coupling of Heterogeneous Embedded Systems," *RCBS Workshop on Model-Driven Embedded Systems 04*, 2004.

[5] Elena Litani and Ed Merks and Dave Steinberg, "Discover the Eclipse Modeling Framework (EMF) and Its Dynamic Capabilities," *Website*, 2008. [Online]. Available: http://www.devx.com/Java/Article/29093

[6] MetaCase, "MetaEdit+®Workbench and Modeler," Website, 04 2008. [Online]. Available: http://www.metacase.com/mwb/

[7] Detlef Streitferdt and Philipp Nenninger, "Quality Assurance Challenges in the Industrial Automation Domain," in *Business Process Engineering (CONQUEST proceedings)*. Dpunkt Verlag, 2007.

[8] Thomson Haydn, "Flexible Control Systems Development and Integration Environment for Distributed Systems," *ATP International, Engineering Embedded Systems*, 2007. [Online]. Available: http://www.atp-international.de

[9] Inchron, "Real-time simulator chronSim," *Website*, 2008. [Online]. Available: http://www.inchron.de

[10] Esterel Technologies, "Scade Suite (TM) and IEC 61508 - Certified Code Generation)," *Website*, 2008. [Online]. Available: http://www.esterel-technologies.com/

Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International , pp.1372-1375, July 28 2008-Aug. 1 2008
doi: 10.1109/COMPSAC.2008.156