Softw Syst Model (2013) 12:619–639
DOI 10.1007/s10270-011-0219-7

REGULAR PAPER

Where does model-driven engineering help? Experiences from three industrial cases

Parastoo Mohagheghi · Wasif Gilani · Alin Stefanescu · Miguel A. Fernandez · Bjørn Nordmoen · Mathias Fritzsche

Received: 20 September 2010 / Revised: 3 September 2011 / Accepted: 29 September 2011 / Published online: 26 October 2011

Abstract There have been few experience reports from industry on how Model-Driven Engineering (MDE) is applied and what the benefits are. This paper summarizes the experiences of three large industrial participants in a European research project with the objective of developing techniques and tools for applying MDE on the development of large and complex software systems. The participants had varying degrees of previous experience with MDE. They found MDE to be particularly useful for providing abstractions of complex systems at multiple levels or from different viewpoints, for the development of domain-specific models that facilitate communication with non-technical experts, for the purposes of simulation and testing, and for the consumption of models for analysis, such as performance-related decision support and system design improvements. From the industrial perspective, a methodology is considered to be useful and cost-efficient if it is possible to reuse solutions in multiple projects or products. However, developing reusable solutions required extra effort and sometimes had a negative impact on the performance of tools. While the companies identified several benefits of MDE, merging different tools with one another in a seamless development environment required several transformations, which increased the required implementation effort and complexity. Additionally, user-friendliness of tools and the provision of features for managing models of complex systems were identified as crucial for a wider industrial adoption of MDE.

Communicated by Prof. Perdita Stevens.

P. Mohagheghi (✉)
SINTEF, and Norwegian University of Science and Technology, Oslo, Norway
e-mail: parastoo.mohagheghi@sintef.no

W. Gilani · M. Fritzsche
SAP Research Center, Belfast, UK
e-mail: wasif.gilani@sap.com

M. Fritzsche
e-mail: mathias.fritzsche@sap.com

A. Stefanescu
University of Pitesti, Pitesti, Romania
e-mail: alin.stefanescu@upit.ro

M. A. Fernandez
Ericsson, Valladolid, Spain
e-mail: miguel.a.fernandez@ericsson.com

B. Nordmoen
WesternGeco, Asker, Norway
e-mail: nordmoen@slb.com

Keywords Model-driven engineering · Domain-specific language · Simulation · Experience report · Eclipse · Complex systems

Abbreviations

API	Application Programming Interface
BPM	Business Process Modelling
DSL	Domain-Specific Language
DSML	Domain-Specific Modelling Language
GUI	Graphical User Interface
MBT	Model-Based Testing
MDE	Model-Driven Engineering
MDPE	Model-Driven Performance Engineering
OMG	Object Management Group
SOA	Service-Oriented Architecture
UML	Unified Modeling Language

1 Introduction

For years, modelling has been applied as an important part of software development in order to tackle complexity by providing abstractions of a system and hiding technical details. Nowadays, industry tends to use models more and more for tasks other than just describing the system, for example simulation, generating source code and performing various analyses at the model level. Model-Based Software Development (MBSD) refers to approaches that use models extensively in software development and covers a wide spectrum of approaches, from using models for some generation to "models only" [1]. Model-Driven Engineering (MDE) is an ambitious goal of using models in all the stages of software development and performing transformations from models to other models or text such as source code. MDE is a new paradigm in software development that promises to solve several problems faced by developers by raising the abstraction level and introducing more automation in software development. In fact, significant improvements have already been reported by some industry, as discussed in [14]. Domain-Specific Modelling (DSM) approaches have also long been practiced by industry to create modelling languages and generators and we find several successful cases reported in the series of workshops on DSM and by the DSM forum.[1] On the other hand, other companies have not yet started to apply model-driven approaches due to the associated cost and risks (heavy changes to the software development process are required), the lack of expertise, "immature" tools, or the lack of insight into the contexts in which the approach can give useful results.

This paper presents experiences from the EU FP6 research project MODELPLEX[2] (September 2006–February 2010), in which large industrial partners applied MDE on different scenarios of complex software systems' development. The industrial partners were from different domains, i.e. enterprise business applications, telecommunication, airport crisis management systems and data-intensive geological systems. MODELPLEX had a total number of 21 partners, among them tool vendors, research organizations, academia and consultancy companies based in eight countries. By putting together tool providers and large industrial users, the project provided a ground to evaluate the state of the practice of MDE, its benefits and pitfalls, and to improve the state of the art. The three industrial cases represented in this report involved more than 30 person-years of effort and thus the experiences are from large-scale cases spanning several years.

The remainder of this paper is organized as follows: Sect. 2 presents the research context while Sects. 3–5 are the description of the three industrial cases, how MDE was applied and gained experiences. Section 6 provides summary, discussion and a comparison to the state of the art. Finally, the paper is concluded in Sect. 7 and gaps for future research are discussed.

2 The research context

Complex software systems have a strong need for a model-based development approach since levels of abstraction are required in order to enable human comprehension, communication and analysis, as well as for the synthesis of implementation artefacts. Complex systems in MODELPLEX were described using five complexity dimensions, i.e. *size*, *heterogeneity*, *distribution*, *dynamicity* (requiring re-configuration and management) and *autonomy* (loosely coupled, cooperating and autonomous systems). To address the challenges of developing such complex systems, MODELPLEX identified the need for innovations along four research areas:

- *Model engineering capabilities* for developing methodologies and tools for modelling complex systems. These capabilities covered Domain-Specific Modelling Language (DSML) engineering, composition of models, the derivation of models from legacy code, and traceability between requirements and models, between different models and between models and generated artefacts.
- *Verification and validation capabilities* covering model verification engines, performance analysis, simulation tools, test generators and model debugging tools.
- *System management capabilities* to manage and configure models after deployment, especially for systems of systems and highly distributed devices.
- *Common software architecture* for integrating the MODEDPLEX tools and techniques in an Eclipse-based "MODELPLEX platform". The MODELPLEX project also offered a lightweight model repository and model management facilities for managing large models. Every industrial partner took advantage of a set of MODELPLEX tools in addition to other external and internal tools, integrating these in a case-specific *workbench*. Examples are provided in the description of cases.

Research in all these areas included developing metamodels, model transformations and defining integration or mapping solutions to other tools.

Several of MODELPLEX tools were developed as open source projects, mostly as Eclipse plug-ins. Also a number of extension points were defined to support the plugging of different implementations providing similar functionality. For example, all model-based testing tools can use the defined extension point for plugging into the platform. The project did not provide a UML modelling tool and the MODELPLEX platform can be configured with a UML modeller of choice that is Eclipse based and uses the standard UML2

[1] http://www.dsmforum.org/cases.html.

[2] MODELPLEX (MODelling solution for comPLEX software systems); http://www.modelplex.org.

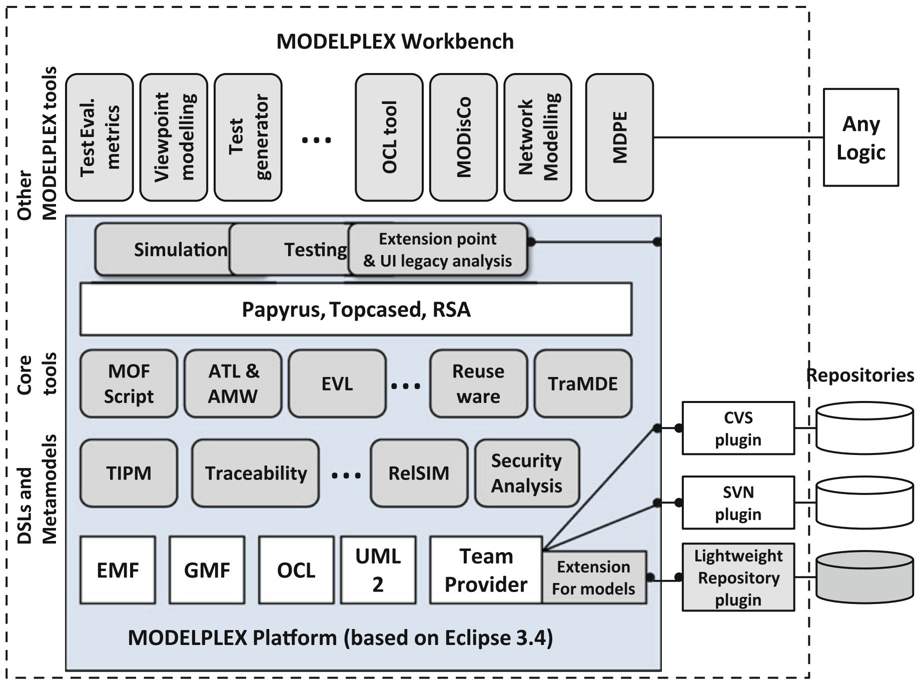


Fig. 1 MODELPLEX architecture

technology. Examples of modelling tools that satisfy these requirements are Papyrus[3], Topcased[4] and Rational Software Architect (RSA).[5] However, the practice showed that integrating MODELPLEX tools with different modelling tools is not a straightforward task and there are several problems in exchanging models as discussed later.

Figure 1 shows the MODELPLEX architecture, including metamodels, extension points for simulation/testing/legacy analysis tools, some of the tools developed within the project (in grey colour), other frameworks and tools being counted as external tools (in white colour), and the AnyLogic[6] simulation tool which is a commercial tool from XJ Technologies who was a MODELPLEX partner. Appendix includes a short description of the MODELPLEX tools and their application areas.

MODELPLEX aimed to drive the research from the needs of industrial partners. At the beginning of the project, the industrial partners defined a set of requirements within the context of their cases that reflected their business goals, objectives and needs regarding MDE technologies and tools (generally called *solutions*). These requirements were the basis for developing solutions by tool vendors and research partners. MODELPLEX was also strongly focused on the empirical evaluation of the results. In order to evaluate the solutions in a systematic way, all industrial partners defined case-specific *research questions* that reflected their goals with using MDE and their criteria for evaluation. Research questions were answered by performing empirical studies during the project such as case studies and internal surveys. The empirical evaluation method in MODELPLEX is discussed in detail in [15].

In the following sections, we describe the context of research performed in three industrial cases and focus in each case on research areas that produced interesting results. We also use experiences from other research performed in the project to understand and interpret the findings.

3 SAP: large-scale enterprise business applications

SAP's technology platform is based on a Service-Oriented Architecture (SOA). This architecture elevates the design, composition and management of enterprise services. The SAP case is regarded as complex in the dimensions of size and distribution, as the target systems are enterprise-scale business processes which are inherently distributed and potentially composed of hundreds of enterprise services. Different technologies can be used to develop the applications providing and consuming enterprise services while these services are integrated by the concept of "Composite Applications (CA)" [6]. Thus a CA is realized as a composition of existing services as well as third party services. The main challenges that were addressed during MODELPLEX in the context of CA were

(a) Predicting performance at the design time and continuously optimizing the performance at run time. The goal

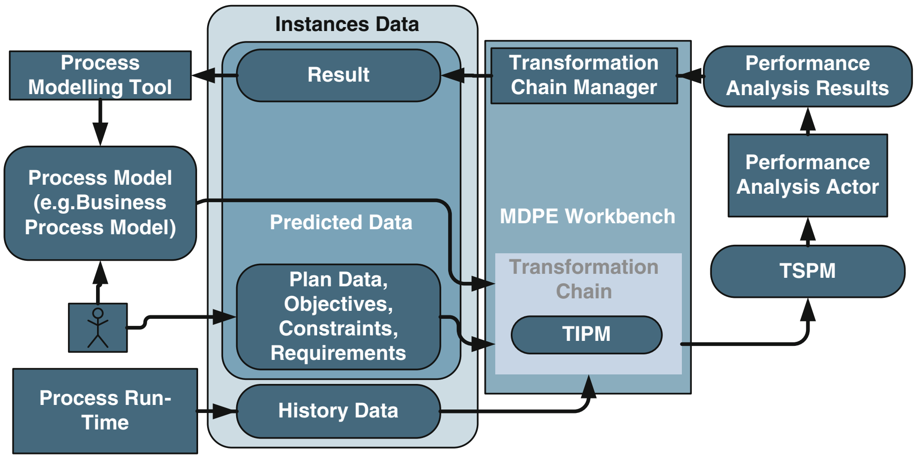
[3] Papyrus; http://www.papyrusuml.org/.

[4] Topcased; http://www.topcased.org/.

[5] RSA;http://www.ibm.com/software/awdtools/architect/swarchitect/.

[6] AnyLogic simulation tool; http://www.xjtek.com/.

Fig. 2 MDPE workbench


Instances Data
Process Modelling Tool
Result
Transformation Chain Manager
Performance Analysis Results
Process Model (e.g.Business Process Model)
Predicted Data
MDPE Workbench
Performance Analysis Actor
Transformation Chain
TSPM
Plan Data, Objectives, Constraints, Requirements
TIPM
Process Run-Time
History Data

was to support business experts, who generally lack performance-related skills, to take decisions related to the performance of business processes, for example when resources change.

(b) Integration testing of the components involved in a CA.

SAP is already advanced in MDE technologies using it on a daily basis in its software development [13] and SAP's use case in the project concentrated primarily on exploring ways to extend existing MDE tools and processes with new capabilities in two dimensions, i.e. *Model-Driven Performance Engineering* (MDPE) and *Model-Based Testing* (MBT). The work on performance analysis (MDPE) is described below in more detail.

The list of tools used in this case cover

- Tools developed and extended within the MODELPLEX project:
  - MDPE workbench by SAP
  - IBM's Test generator tool
  - FOKUS!MBT: Test derivator tool
  - MBTM: Model-Based Testing Metrics tool
  - AnyLogic: commercial simulation tool
  - AMW: Atlas Model Weaver tool
  - ATL: Atlas Transformation Language
- Tools not part of the MODELPLEX project:
  - RSA as the modelling tool
  - SAP NetWeaver Business Process Management (NW BPM) Suite
  - JCOM BPM suite[7]
  - SAP NetWeaver Developer Studio
  - SAP Modelling Infrastructure (MOIN)

3.1 Research method

The empirical research method was mainly an exploratory case study of a real business scenario called "Sales Order Management System". Business process models of this system were utilized to generate performance analysis models and test models. The SAP Research team within MODELPLEX worked very closely with the SAP's internal departments that deal with the development of MDE-related tools and technologies. These departments provided the authors with real-use cases and were involved in the evaluation process. The evaluators were therefore the SAP internal product development departments. The cooperation was formalized in five internal transfer projects in the past 3 years. These projects were meant for transferring research results into internal SAP product development departments. The deliverables of these transfer projects have been the basis of our evaluation work as described in detail in [7,17]. The participants in these projects were able to interactively apply the changes in the business process models and observe the results. The variants of the new business processes were adapted to answer different research questions.

3.2 Model-Driven Performance Engineering (MDPE)

Figure 2 shows the abstract architecture of the MDPE workbench. The existing business process models and the process history data available from the Business Process Management tools are transformed into analysis models in an automated fashion via a model transformation chain. These analysis models are then fed into the Performance Analysis Actor, which could be any analytical tool of choice such as a simulation engine (in this case the AnyLogic tool) or an optimization tool (e.g. FMC QE[8]). The analytical tool does

[7] http://www.jcom1.com/.

[8] http://www.hpi.uni-potsdam.de/zorn/projekte/fmc_qe.html.

Table 1 Transformation complexity comparison: transformation chain versus monolithic approach

Transformation	LOC	(#FunctionCalls + #HelperCalls + #Conditions)/LOC
BPMN2UML	166	0.0542
UML2Anylogic (via TIPM and others)	3,485	0.2998
Monolithic (BPMN to AnyLogic)	1,865	0.1426

the computation and produces the results, which are brought back and annotated on the original business models. Business domain experts are therefore able to click a button from their Business Process Modelling (BPM) environment, invoke an analysis such as simulation and see the results decorated on the business models that they are familiar with. During the process, the following steps are performed in the following background:

1. Business process models are annotated with performance-related data, e.g. objectives related to execution times or resource demands and constraints. A special model annotation technique based on ATLAS Model Weaver (AMW[9]) allows annotating proprietary models without the need to have access to or extend their metamodels.
2. The MDPE workbench is required to support a number of different modelling languages (for example, different proprietary languages used within SAP) as well as various analysis simulation, optimization and analysis tools. Therefore, an intermediate metamodel named *Tool-Independent Performance Model* (TIPM) is introduced in the transformation chain as an abstraction for performance analysis tools to make the solution generic and reusable. TIPM is a representation of the system from the performance perspective. Besides TIPM, UML activity diagrams are used as another intermediate model for abstracting different Process Modelling Languages. The first transformation is therefore always from process models (e.g. SAP proprietary modelling languages) to UML activity diagrams which are then transformed into a TIPM in another transformation step.
3. The TIPM is thereafter transformed into Tool-Specific Performance Model (TSPM) in order to be consumed by an analytical tool of choice.
4. The TSPMs are fed into the analysis engine and the results are collected.
5. The results are finally traced back to the original business models by using the automatically generated trace models from ATL HOT (Higher Order Transformations), which is a way to generate new transformers using existing ones.

The model transformation chain was required as a measure for modularisation, since we were required to hide complexity at the design-time when a new performance analysis tool or process modelling tool is integrated. For instance, if a monolithic transformation is implemented, a transformation developer needs to learn two new concepts, namely the process modelling language and the performance analysis input format. This is normally not the case if the TIPM is employed. Provided that the process modelling language is already translated into the TIPM and the transformation developer has already implemented a translation between TIPM and another Performance Analysis Tool, only the concepts of the performance analysis input need to be learnt.

This leads to the claim that the development of monolithic transformations is more expensive in terms of the development costs if more than one Performance Analysis Tool is integrated with more than one Process Modelling tool.

We have measured the development costs based on the following two metrics (see Table 1):

- The second column of Table 1 shows the number of lines of ATL code (LOC) needed for the transformation (steps) of the transformation chain case study for a transformation via the MDPE chain versus the comparative monolithic transformation.
- In the third column of Table 1, a metric for the average complexity for each LOC of the single delta transformation from BPMN to the UML Activity Diagrams and the comparative monolithic transformation from BPMN to AnyLogic is provided. This complexity metric is calculated based on the number of helper calls, conditions and ATL-function calls per line of ATL code. This metric is based on the function metrics to measure certain quality attributes of model transformations proposed in [18].

If BPMN models are employed, only 166 additional lines of low-complexity code are added (see row 2 of Table 1), which is of significantly lower complexity than the monolithic transformation (see row 4 of Table 1). Thus, low effort is needed which is partly a result of the fact that the BPMN language (and many other languages which are similar to BPMN) can be easily translated to UML Activity Diagrams. However, these numbers consider that the remaining part of the transformation chain, e.g. the translation from Activity Diagrams to AnyLogic via the generic TIPM is already available. The complexity of those intermediate transformation steps is higher than the BPMN2UML transformation and also higher than the monolithic transformation (see row 3 of Table 1). Thus, the transformation chain only pays off when

[9] ATLAS Model Weaver; www.eclipse.org/gmt/amw/.

Table 2 Performance penalties of the generic four-step transformation chain compared to the monolithic approach

Transformation	Memory footprint (Kb)	Performance (s)
BPMN to AnyLogic via the transformation chain	2,274	13.6
BPMN to AnyLogic via a monolithic transformation	595	3.2

several process modelling tools need to be integrated with several performance analysis tools.

The price to pay for this modularisation at the runtime was a memory and runtime overhead. This was due to the overhead for maintaining intermediate models, tracing, etc. To quantify this, we did measurements on the performance of the MDPE workbench. For instance, the time taken to transfer data from a modelling tool to an analysis tool was more than three times higher for the multi-step transformation chain, compared with the monolithic approach, which involved direct transformation from a specific modelling tool to a specific analysis tool. These times were computed for a model describing a business process with 47 high-level steps. The input BPMN model used 202 Kb of memory. However, only 20 Kb of these data were behaviour specific and were needed for the simulation. The remaining information mainly concerned modelled rules. Additionally, 47 Kb of data representing performance parameters were injected into the MDPE transformation chain. The resulting 67 Kb (20 + 47) of data were transformed with the MDPE transformation chain into an AnyLogic model, which used 528 Kb of memory. Thus, the monolithic transformation required only 595 Kb (528 Kb + 67 Kb) of memory to serialize the input and output models. Therefore, the memory footprint for the chain with 2,274 Kb was significantly higher than the monolithic transformation. The results are summarized in Table 2. All transformations were implemented with ATL. For the measurements, a computer having 2 GB of RAM and a 2 GHz Dual Core CPU was employed.

3.3 Conclusions

The MDPE workbench enables non-intrusive integration of sophisticated and automated performance-related decision support into existing BPM environments. The workbench was demonstrated to a number of actual and potential large customers of the SAP NetWeaver BPM product. In all these cases, the feedback was largely positive in terms of the usefulness of the approach. Besides this, we also experienced drawbacks and benefits related to the various MDE tools employed within the MDPE workbench as discussed below.

On the positive side

- Generally, we experienced that dealing with data conforming to a metamodel is a powerful concept for building applications with complex data structures and input/output languages. This is mainly caused by the Eclipse Modelling Framework (EMF) and its code generation capabilities, reflective API, documentation, user groups, etc.
- After a warm-up phase with the ATL language, we found that the combination of declarative and imperative concepts for defining transformations of complex data structures is an efficient approach.
- Also model weaving was straightforward for defining links between our models. For instance, we applied AMW to annotate additional information to the models without polluting the metamodels. The annotations were defined in separate models. This approach can now be applied to annotate DSLs that do not have a profile mechanism like UML.

On the negative side

- The MDPE workbench required an integrated use of Java coding with ATL transformation scripts, tracing models and weaving models. We encountered a lack of integration between these artefacts. It would, for instance, be beneficial to be able to declare transformation rules embedded in the Java coding and runtime. If this had been available, we would have been significantly more productive in implementing the MDPE workbench.
- ATL did not prove to be a very stable tool and also required high expertise to use it. There have been additional issues when we migrated to the latest version of the ATL (ATL 3.1.0 release candidate 4), due to changes in the ATL libraries that are used to read the source models. This required re-coding to make it work again. The absence of debugging support additionally made it difficult to locate errors. We are however aware of the fact that debugging support for rule-based languages is difficult to realize as the programmer does not define an order in which rules are executed.
- When employing modularisation concepts on our model transformations, we missed several concepts and tools. For instance, a systematic concept for managing model annotations and tracing across the transformation chain had to be investigated and implemented by ourselves. Moreover, our implementation showed a significant runtime footprint as described above. Related to this, we propose two directions for future research which should decrease both the runtime overhead of modular transformations and the need to develop such decomposed transformations again and again:

(i) Support for transformation chains could be part of the transformation tool and the management of intermediate models and tracing should be done within the framework in a performance-optimized way.
(ii) Moreover, transformation chains at runtime can be avoided by merging model transformations, similar to program optimization approaches like deforestation [19]. We explained this idea in more detail in [8].

As a simulation engine at the end of the transformation chain, AnyLogic tool was used. Being commercial, the tool is quite stable and mature. However, there was a technical integration issue because AnyLogic did not offer some of the APIs without the GUI. Our requirement was to have the tool running only in the background. Open source tools were therefore added as additional analysis engines apart from AnyLogic.

Apart from tool issues, there has been one important concern for the MDPE approach itself, which is its reliance on the existing historical data about the business processes and the business process models themselves. Especially, when dealing with human-centric business processes, we identified that the available historical business process data were sometimes of poor quality. This was either due to the limited amount of data available for certain activities in the process or due to high variance in the data for certain process activities processed by humans. In order to address this issue, we extended the MDPE workbench with an additional tool that analyses and corrects the data before simulation.

We also performed research on MBT which covered transforming models from proprietary domain-specific languages of SAP into UML with Java annotations and UML-2 Testing Profile (U2TP[10]). Test cases were generated using two test generators developed within MODELPLEX and transformed back into SAP internal test case format. The integration of MDPE and MBT workbenches with SAP tools has been evaluated as successful. The research prototype is currently being productized into an industry solution and is considered for inclusion in a future version of SAP BPM tool. The next steps are to address issues like incorporating exceptions in business processes and dealing with the high variance in performance data available from business logs.

4 Telefónica: network modelling in telecommunication domain

Telefónica's case in MODELPLEX used the CPE Management and Configuration System (CMCS), where CPE stands for Customer Premise Equipment. The aim of this system is the automated and remote (when possible) configuration and monitoring of the network devices necessary for broadband service provisioning in Telefónica. The case in MODELPLEX is considered as complex especially regarding size and heterogeneity. Regarding size, we can mention that there can be thousands of configurable elements in a typical network device and thousands of network devices are monitored. The system is heterogeneous since there will be multiple versions of the CMCS system that must be integrated in different Operational Support Systems (OSS), with different services and features offered to the customers.

One of Telefónica's research focuses in MODELPLEX was the development of a DSML and related tools for modelling network devices and services involved, so that faster service development and deployment is possible, by substituting the current practice of textual service specification with a model-centric approach. This case is described in detail below. Before that, the tools used in this case are listed below:

- Tools developed or extended within the MODELPLEX project:
  - Telecom Service DSL: developed as part of this case
  - Reuseware composition framework: used for model composition
  - EuGENia: used for generating GMF editors
  - EVL: Epsilon Verification Language
  - ATL: ATLAS Transformation Language for model-to-model transformation
  - FOKUS!MBT: Test derivator tool
- Tools not part of the MODELPLEX project:
  - EMF: Eclipse modelling and metamodelling framework
  - GMF: Eclipse framework and tooling for DSL definition
  - RSA, Papyrus and Topcased: UML2 modelling tools

4.1 Research method

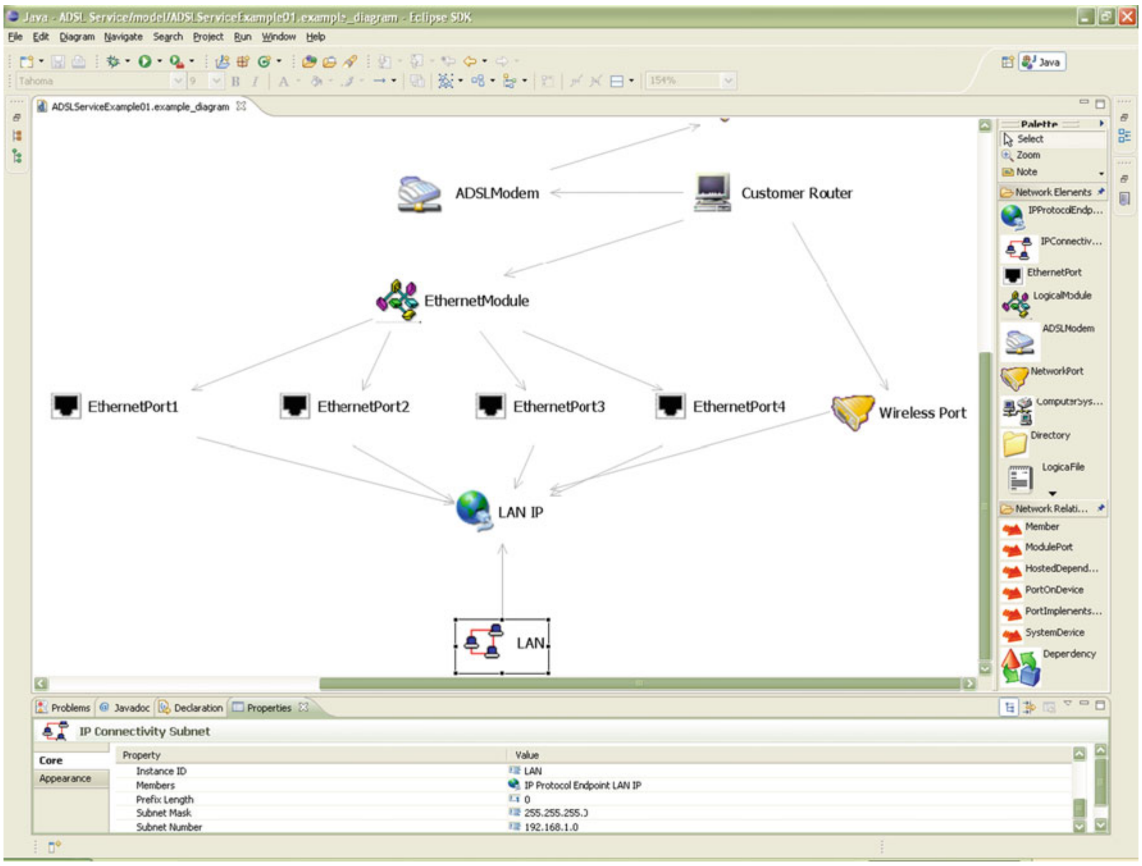
Our research was mostly performed as case studies with comparison between MDE and other approaches. Additionally, we conducted several internal surveys to collect the opinion of the developers participating in case studies about the new technologies and their perception of the usefulness of MDE in general. The development of solutions has been iterative and is best described as research-in-progress.

4.2 The telecom DSL prototype

We started, early in the project, working with project partner Xactium[11] in the development of a first prototype and proof

[10] http://utp.omg.org/.

[11] Xactium; http://www.xactium.com.



IP Connectivity Subnet

Property	Value
Instance ID	LAN
Members	IP Protocol Endpoint LAN IP
Prefix Length	0
Subnet Mask	255.255.255.)
Subnet Number	192.168.1.0

Fig. 3 First proof of concept of the CIM-based telecom DSL tool

of concept DSML tool using the Eclipse GMF. Experiences with developing this DSML are discussed in detail in [3]. A metamodel based on Common Information Model (CIM)[12] was used in this development. CIM is a recognized industry standard and it is widely used in instrumentation software and many off-the-shelf products for network management. The overall GUI of the first prototype can be seen in Fig. 3.

The evaluation of this first tool by several engineers in the CMCS development group was unanimous. The work was deemed as very promising and the selection of CIM as the underlying metamodel was considered as a very pertinent solution. However, some critical areas for improvement were also identified:

- The need for a much more powerful GUI that allows telecom domain experts to efficiently create domain-specific models with a huge number of meta-classes available (CIM has more than 1,500 elements). For this reason we decided to scrap the first prototype and design a new, more powerful, GUI.
- The need for modelling at a higher level of abstraction and developing hierarchical models triggered using model composition tools such as the Reuseware composition framework developed in MODELPLEX.

The most negative aspect of this prototype DSML tooling has been the difficulty of modifications. The telecommunication domain is very dynamic and this poses a big challenge in keeping the tool updated. This aspect of the tool (being able to keep up with metamodel variability) is not satisfactory in the current implementation. GMF lacks the necessary characteristics to cope with such a quick development need. Every time the metamodel changed, which happened several times during the development of the tool, a lot of manual work was needed to update the tool. However, a new framework developed in MODELPLEX called EuGENia facilitates generating GMF editors. EuGENia automatically generates the .gmfgraph, .gmftool and .gmfmap models needed to implement a GMF editor from a single annotated Ecore metamodel and thus hides the complexity of working with GMF. Another problem is that, after the new metamodel and tool are finished, there is still a lot of effort involved in making the old models usable with the new tool.

[12] Common Information Model is a standard developed by the Distributed Management Task Force; http://www.dmtf.org/standards/cim/.

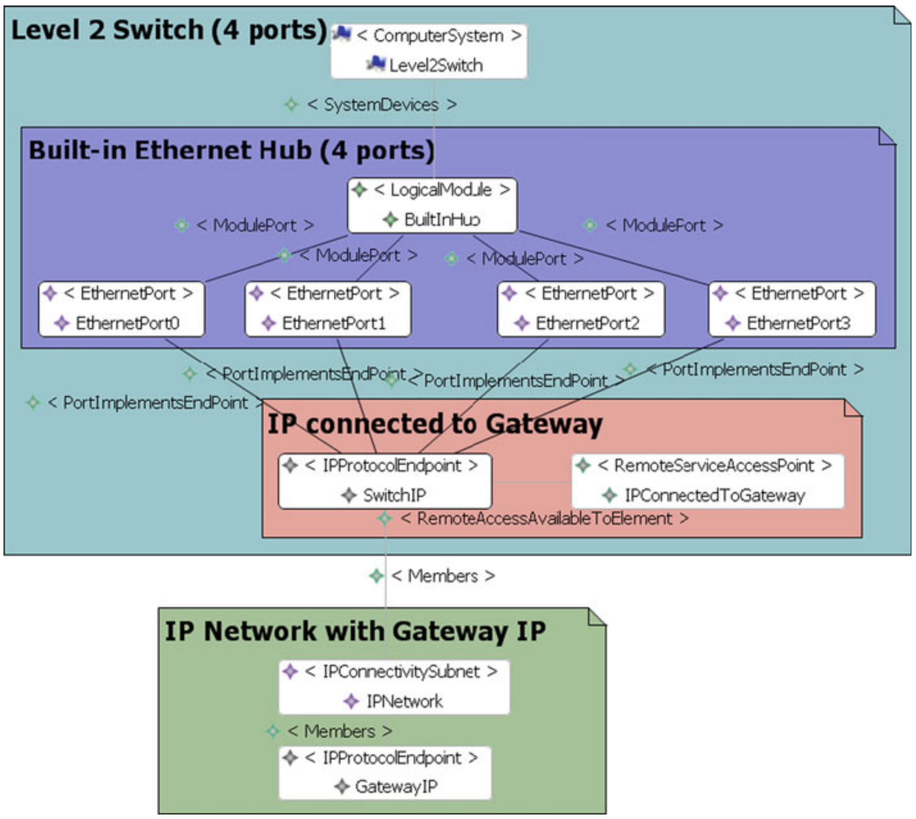


Fig. 4 Manual fragment identification by a domain expert over a simple domain model

The developed DSML was later extended in several ways. Firstly, it was integrated with the verification language EVL for the incremental checking of the correctness of the models being created. EVL was evaluated by our experts as very beneficial in terms of usability, as it allows the incremental checking for correctness of the models while being created. A set of rules from our domain was documented and then implemented in EVL. Where appropriate, quick-fixes were implemented as well. Second, to support modelling at higher levels of abstraction than single network elements, a composition system was developed using the Reuseware tool [10]. This multi-layer DSL approach was based on the domain concepts identified in the previous work. At the higher levels of abstraction, model fragments with specific domain meaning are composed. These identified fragments, together with the proper semantic information are then stored in a fragment repository where they can later be searched and reused in other models. One example of this process of fragment identification, done by our in-house domain experts, can be seen in Fig. 4. The semantic tags, or "facets", allow the characterisation of the fragments in the repository regarding the structure of the fragment, viewpoint (e.g. physical, logical, software, etc.), or domain properties (e.g. network protocols supported, etc.). In terms of graphical customization, we worked on hiding the parts of the composition program that are not relevant to the modeller (the user) and in providing graphical representations for the different fragments that are stored in the fragment repository.

An alternative approach to handle modelling at several levels of abstraction would be to develop multiple DSLs and define transformations between them, which could be simpler for the end-user since there is a fixed level of abstraction. The composition approach is, on the other hand, more flexible but the recursion levels of compositions can get too complex. To better restrict these options, we proposed an iterative process for defining the multi-layered DSML environment, as also described in [16].

Other than modelling, the purpose of our DSL was to generate network device configuration files out of our domain-specific models. We experimented with the ATL transformation language and for comparison we developed the same transformation chain using Java. The idea was to compare both approaches in the terms of ease of use, performance, readability of the code and ease of learning. A team of eight developers was involved in the evaluation. At the beginning, they preferred Java because of their previous knowledge of the language. By the second evaluation, most of them recognised that ATL had an advantage in the ease of use and the readability of the scripts, even though the learning curve for the language was deemed as high. The performance of ATL was acceptable even for large models. The work with ATL was not free from problems, though. For example, the inability of ATL to manage metamodels contained in more than one Ecore file forced us to implement some workarounds.

4.3 Conclusions

We performed extensive work on model-based testing as well using the Fraunhofer FOKUS test derivator tool FOKUS!MBT and TTCN-3 test execution environment (TTWorkbench, a commercial tool from Testing Technologies). The behaviour models best suited to our purpose were found to be UML state charts. However, the test derivator tool did not have adequate support for generating test cases from UML2 state charts and was therefore extended in MODELPLEX. The Fraunhofer partner also had to customize their tool for different UML2 modelling tools with non-compliant XMI outputs. We experimented with different modelling tools and we chose RSA in the end. Our main requirement for the evaluation was the overall productivity of the testing process so that less effort is spent on test scripting, while test coverage is satisfactory. The initial results were considered as satisfactory but only after the extensive work on modelling and tool customization. Since test cases and TTCN-3 test scripts were automatically generated from the models both goals can be achieved: the user has a good control of test coverage via the test models and less effort is spent on test scripting because the test scripts are automatically generated and not written by hand.

We had both positive and negative experiences with the tools. On the positive side, most tools that we evaluated (such as ATL and Reuseware) are close to maturity and are basic and helpful technologies that provide a good support to very diverse MDE approaches. On the negative side

- We experienced that a high level of expertise is required for developing editors, customizing tools and developing workarounds. In this project, we received direct help from tool vendors.
- We also observed that no single modelling tool could serve all of our purposes. In our case, we decided to use IBM's RSA as our UML2 modelling tool, which is a licensed tool that covered most of the functionality for our case. Early models created in Papyrus and some other tools had to be re-done from scratch because of the issues with exporting/importing to XMI between tools. Diagrammatic information is not preserved from one tool to another and there is incomplete implementation of the UML2 specification in most tools.

Our goal for participating in the project has been to study whether the MDE approach can be useful in our context with the current state of the art and if yes, in which scenarios. With the current state of the art, we have found some interesting uses for the approach, like our experimentation with hierarchical DSLs and our work on model-based testing. One observed benefit is that models can act as documentation and no extra documentation is needed to understand the modelled networks.

The DSL development work is considered as very promising. However, we have reached the conclusion that DSLs should be smaller to be practical. Since CIM metamodel contains over 1,500 concepts, a reduced subset relevant for the case was used in the first implementation, consisting of more than 200 concepts. There are also a large number of relations in the CIM metamodel. We applied techniques such as modelling concepts as nodes and edges in a graph and reducing the number of palette elements by grouping classes. For the future, either the relevant subset should be reduced even more or the grouping should be enhanced. As also emphasized in [12], a language with too many concepts creates problems during language deployment and use.

We have, however, our doubts about whether MDE can be a complete alternative to a more traditional development paradigm because of the state of the tools that create several obstacles in applying the approach. Problems with customization and interoperability between modelling tools were discussed above. Additionally, we observed that traceability in the life cycle is difficult to manage when putting together several tools and MODELPLEX could not deliver a satisfactory solution for a model repository. In conclusion, we have observed many benefits of the MDE approach such as

- Domain models are easier to understand for those who deal with the configuration of nodes;
- We can achieve reuse and generate documentation from models;
- Model-based testing has the potential to save effort.

However, developing a MDE tool chain requires high expertise and an investment of effort since no platform works "out-of-the-box". It will pay off if we can convince the company for large-scale reuse of the solutions, which requires more experimentation and increasing our expertise, in addition to more stable and interoperable tools.

5 WesternGeco: model-based simulation and testing

WesternGeco (WGO) works in the area of Oil and Gas Exploration, offering advanced seismic surveying services. The aim of seismic surveying is data acquisition to produce images of geological features and their structure below the surface of the earth. The system under focus in MODELPLEX was the onboard Spread Management System, which will boot up, control and monitor the spread

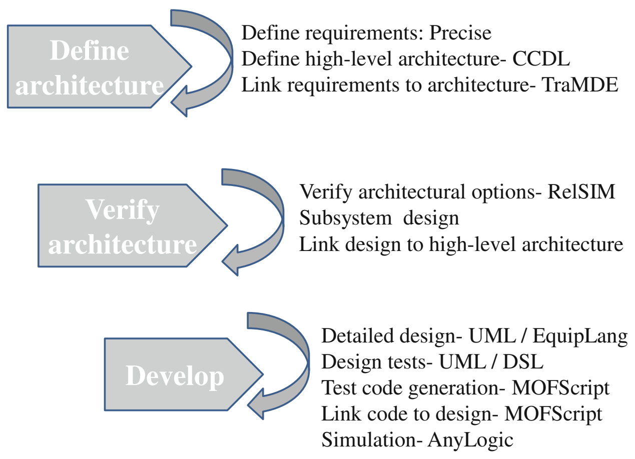

Define architecture
Define requirements: Precise
Define high-level architecture- CCDL
Link requirements to architecture- TraMDE
Verify architecture
Verify architectural options- RelSIM
Subsystem design
Link design to high-level architecture
Develop
Detailed design- UML / EquipLang
Design tests- UML / DSL
Test code generation- MOFScript
Link code to design- MOFScript
Simulation- AnyLogic

Fig. 5 MDE development process and tools in the WGO case

instrumentation. The spread instrumentation is a complex network of distributed computers, various devices and sensor nodes. The system is used to perform seismic surveys at sea in order to find oil or gas beneath the sub-sea surface. The case in MODELPLEX is considered as complex, especially regarding size and distribution. The system will contain more than 1,000 distributed computers running embedded Linux. The data-rate back to the onboard instrument-room will be more than 200 MB/s, and the onboard control and processing system will contain several hundreds of cluster-nodes.

In order to secure quality at the right cost for such complex systems, there are two key areas where MDE can make a significant contribution:

(a) It is very important that the architecture of the system (network topology and redundancy) is correct before the system is put into mass-production. Developing architectural models and simulating them will reduce the risk of making wrong design decisions.
(b) Embedded systems contain both hardware and software. In order to avoid forcing software development to wait until the hardware is available for testing, and also because a test setup involving all the real hardware will be very costly, it is important to be able to test the software using hardware emulators. An MDE approach based on modelling the external hardware and using the generated code in testing is expected to produce more reliable tests than hand-coding these tests.

Figure 5 below gives a first cut of the MDE process that supports the two goals described above. The WGO tool suite contained the following tools:

- Tools developed and extended within the MODELPLEX project:
  - EuGENia: GMF front-end
  - MOFScript: model to text transformation tool
  - RelSIM: DSL for reliability modelling and transformation to AnyLogic developed by WGO
  - EquipLang: DSL/UML profile for black-box streamer testing developed by WGO
  - CCDL: DSL/UML profile for architecture modelling developed by WGO
  - AnyLogic: simulation tool
  - TraMDE Traceability framework and plug-in for Precise for requirement to model traceability
- Tools not part of the MODELPLEX project:
  - Eclipse tool suite framework, EMF and GMF
  - Papyrus and Together Architect: modelling tools
  - Precise: requirements management tool developed by WGO

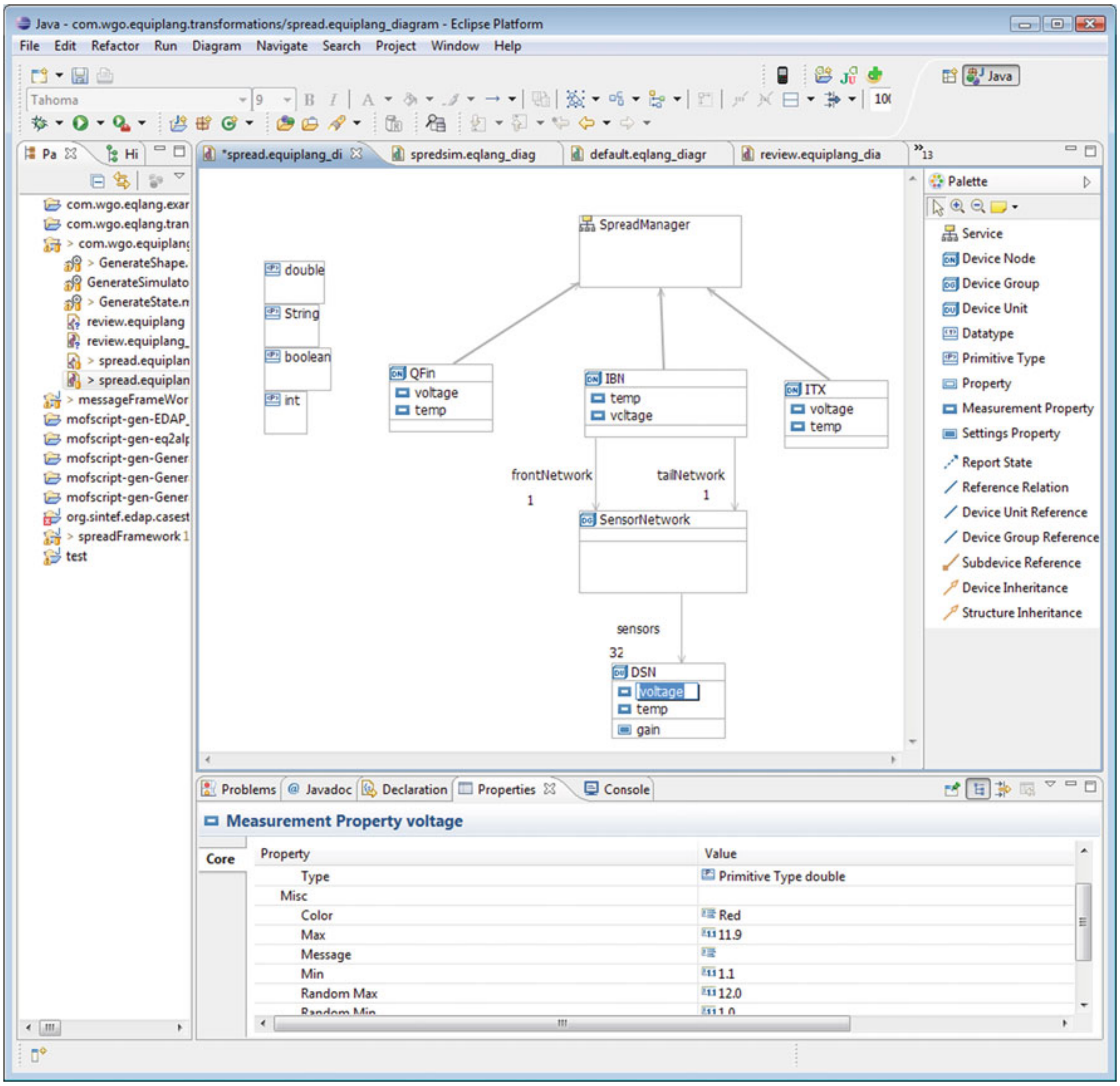
To model software architecture of the spread management system, an architectural DSL was developed (CCDL: Connector Component Description Language). Architectural modelling is supported by the EuGENia tool, which facilitates developing a DSL in GMF. Models developed in this DSL replace the PowerPoint slides of the system architecture that were used before. A key feature of this DSL is the support for drilling down from high-level abstract models into more detailed models, for example to a UML model previously developed. Furthermore, to support simulation of architectural options, a DSL for reliability simulation of the seismic spread equipment was developed (RelSIM) using AnyLogic simulation tool modelling capabilities.

Regarding point (b) above, to ensure that the software is not blocked waiting for the hardware to finish, it is important to develop simulators that emulate the hardware. A DSL for modelling the hardware was developed (EquipLang), and code generators written in MOFScript are used to generate code for the hardware emulators used in the black-box testing of the system under development. We consider this as a successful case of applying MDE to reduce risks in software development and improve the quality of testing. EquipLang was both developed as a DSL and a UML profile for the same domain and we compare modelling and generating code with these approaches in the next section.

The traceability tool TraMDE developed in MODELPLEX is the glue that binds together the different models to one another (from requirements to high-level architecture models and to detailed design models) and to the code.

5.1 Research method

In previous research projects, WGO has created UML profiles for modelling and code generation for a defined platform. Experience has shown that code generation scripts



Measurement Property voltage

Property	Value
Type	Primitive Type double
Misc	
Color	Red
Max	11.9
Message	
Min	1.1
Random Max	12.0
Random Min	1.0

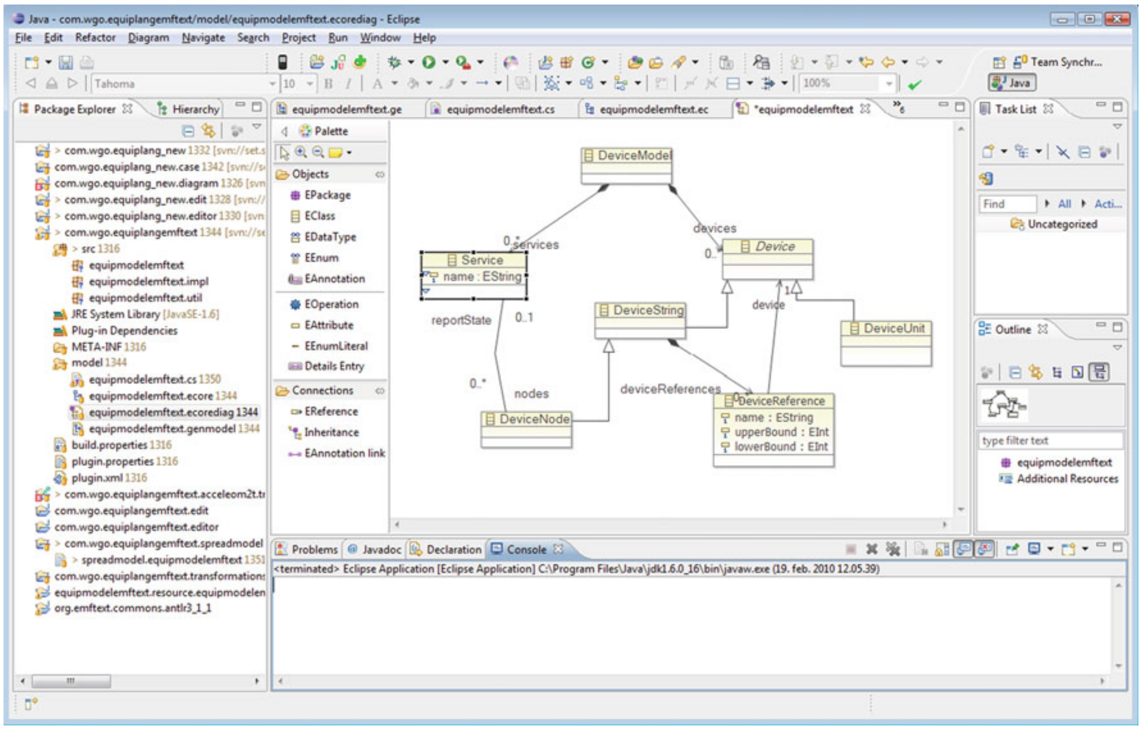
Fig. 6 A spread context model developed by using the EquipLang DSL

become difficult to write due to the complex metamodel of UML. An argument for using a DSL is that things get simpler as the language is tailored to the usage. We performed therefore a comparative study to answer the following questions: first, how easy is it to define a relevant profile in UML using the Papyrus modelling tool versus developing a DSL using Eclipse GMF? Second, how easy is it to write a code generator based on different models? The comparisons here were performed by the team involved in the development of these options.

Regarding developing hardware emulators, simulators were hand-written first and evaluated for correctness by experts. Then code generation scripts were written for generating simulators from the models. The generated simulators were compared with ones that have been hand-written to evaluate their completeness.

5.2 Comparing developing a DSL with a UML profile

Regarding tooling, developing UML profiles is well supported by several modelling tools. We used Papyrus since it is free and allows experimentation at low cost. For developing the DSL, we used Eclipse GMF that was found to be quite complex. Especially, the maturity and intuitiveness of the GMF dashboard are not acceptable. There are two tools developed in MODELPLEX that can be used to simplify the work with GMF: EuGENia and EMFText. Using the EuGENia plug-in, the EMF metamodel can be annotated with GMF tags to simplify the generation of the GMF-based tooling. EMFText is another Eclipse plug-in that allows defining text syntax for languages described by an Ecore metamodel. Figure 6 shows the editor of the developed DSL for modelling the hardware.



```
<terminated> Eclipse Application [Eclipse Application] C:\Program Files\Java\jdk1.6.0_16\bin\javaw.exe (19. feb. 2010 12.05.39)
```

Fig. 7 Streamer context metamodel for the EMFText approach

The main challenge of developing a UML profile is developing a fit for the purpose profile and applying it to the UML language. The UML language in itself is very big and limiting the scope is a challenge. However, the complexity of developing a good DSL should not be underestimated either. Finding the right level of abstraction is not easy and we had to go several rounds, from quite low level to higher level. The lesson learnt here was that the DSL should be at a high level of abstraction; otherwise, you risk reinventing UML. The advantage is that creating the concrete model using a DSL-based editor is much simpler than creating the model using a UML profile since a DSL constrains modelling to those concepts that are needed.

The MOFScript tool was used to develop code generators in both approaches. Writing the code generators were found to be simpler and faster with the DSL since the metamodel did not have the same complexity as the UML profile. We did not collect any metrics on the ease of developing generators, and the evaluation is based on the opinions of the team involved in the research.

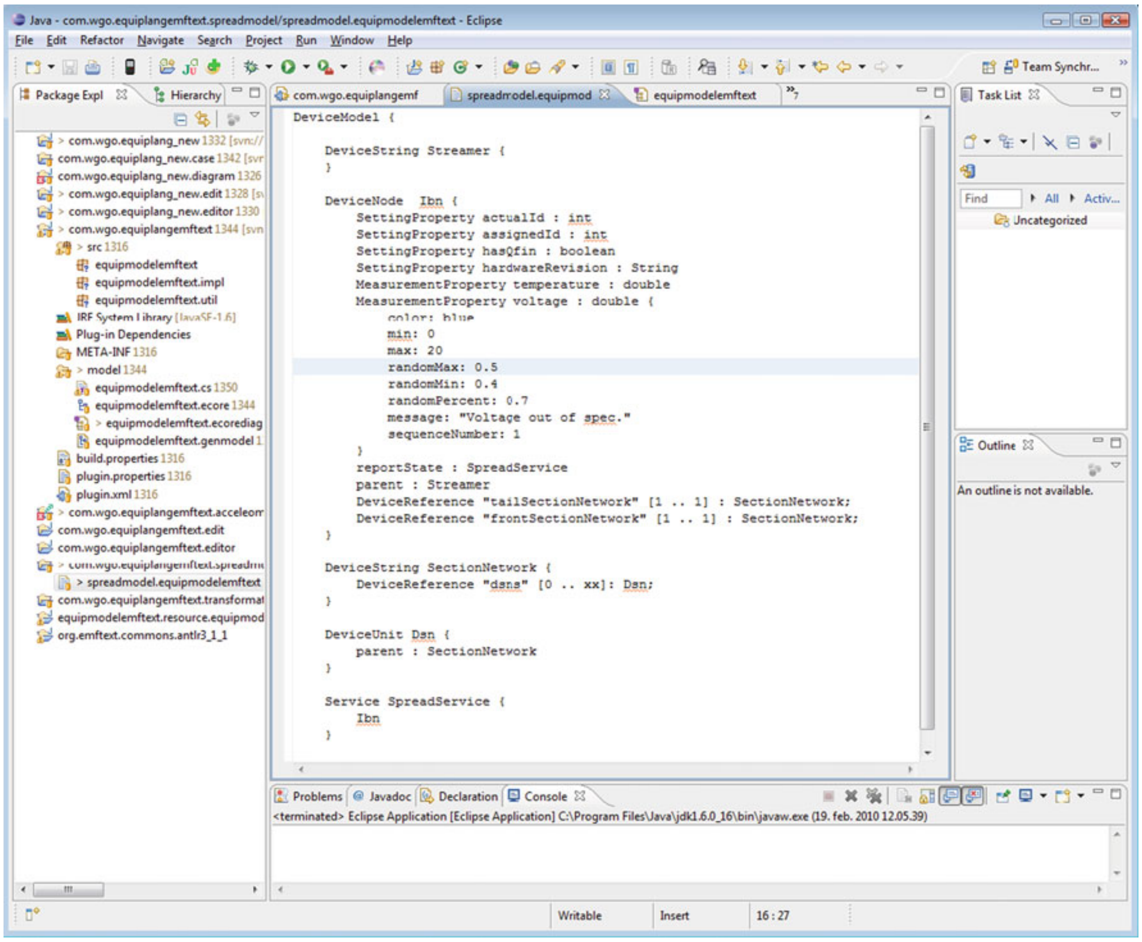
Based on our experience with all the problems using GMF to create DSL tooling, we also tried a third solution using the EMFText tool. The metamodel is pictured in Fig. 7. The tool generates a textual concrete syntax that is depicted in Fig. 8. The language itself is small and easy to use and the code generation with MOFScript was simpler than for UML profiles.

5.3 Testing using simulator generation

When developing systems the project teams typically design and develop simulators in place of the hardware or other software that is either not ready at the time or inconvenient to interface during the development phase. These simulators are used to test the software under development. This test approach implies to develop a *Context model* of the external system (in this case the streamer instrumentation) from which we can generate hardware simulators/emulators to test our system. Our approach was to develop a context model comparing several tools and methods. The code generation was done with MOFScript.

Experiences regarding developing a DSL versus a UML profile were discussed before. An example model developed in the SpreadViewer tool based on the EquipLang is depicted in Fig. 9, which visualizes the state and condition of the streamer hardware. From this tool we can interactively simulate the failure of various instruments and cable cuts and use them for the black-box testing of the system.

The result of this work is still in an experimental phase, but we found that simulating the hardware using models is both feasible and useful. We are therefore following the work in another research project.


Java - com.wgo.equiplangemftext.spreadmodel/spreadmodel.equipmodelemftext - Eclipse
File Edit Refactor Navigate Search Project Run Window Help
Package Expl
Hierarchy
com.wgo.equiplangemf
spreadmodel.equipmod
equipmodelemftext
Task List
Outline
An outline is not available.
Problems
Javadoc
Declaration
Console
Writable
Insert
16 : 27

```
DeviceModel {

    DeviceString Streamer {
    }

    DeviceNode  Ibn {
        SettingProperty actualId : int
        SettingProperty assignedId : int
        SettingProperty hasQfin : boolean
        SettingProperty hardwareRevision : String
        MeasurementProperty temperature : double
        MeasurementProperty voltage : double {
            color: blue
            min: 0
            max: 20
            randomMax: 0.5
            randomMin: 0.4
            randomPercent: 0.7
            message: "Voltage out of spec."
            sequenceNumber: 1
        }
        reportState : SpreadService
        parent : Streamer
        DeviceReference "tailSectionNetwork" [1 .. 1] : SectionNetwork;
        DeviceReference "frontSectionNetwork" [1 .. 1] : SectionNetwork;
    }

    DeviceString SectionNetwork {
        DeviceReference "dsns" [0 .. xx]: Dsn;
    }

    DeviceUnit Dsn {
        parent : SectionNetwork
    }

    Service SpreadService {
        Ibn
    }
```

```
<terminated> Eclipse Application [Eclipse Application] C:\Program Files\Java\jdk1.6.0_16\bin\javaw.exe (19. feb. 2010 12.05.39)
```

Fig. 8 Concrete textual streamer model based on the EMFText approach

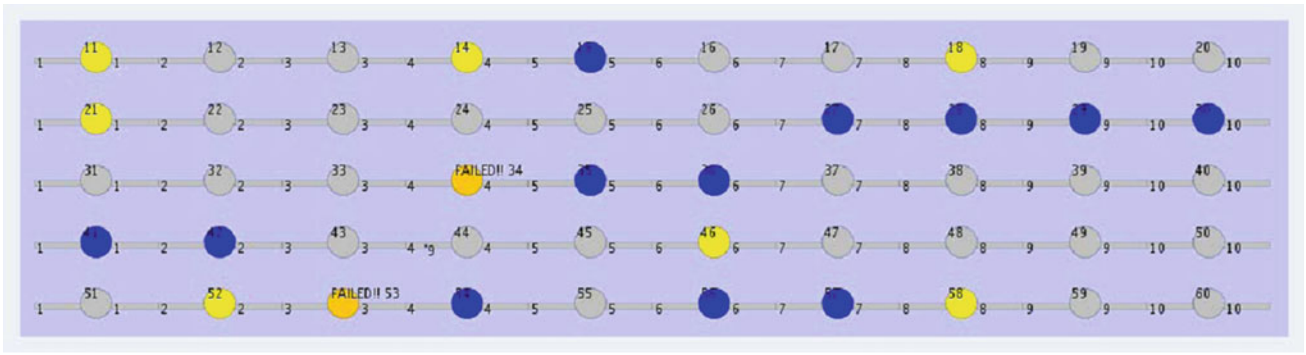


Fig. 9 SpreadViewer visualizing the state of the different instruments in different colours

5.4 Conclusions

We found the tools applied in our research mostly stable and mature:

- AnyLogic, which is a commercial tool, is a complete platform for discrete event- and agent-based simulation and based on MODELPLEX experience, it was introduced and successfully used in WGO in three different internal projects. The only trade-off is the learning curve for AnyLogic, which is a bit steep when simulating complex systems. However, it is worthwhile for complex systems.
- We have been using the MOFScript tool for generating code from models for more than 6 years in different projects and find it useful and stable.
- We also had a positive experience with combining TraMDE and our requirements tool, which allows tracing requirements to models.

- The overall experience is that Eclipse serves well as an integration platform. Talking about usability, GMF tooling showed to be more complex than expected. Using EuGENia and EMFText simplified the generation of DSLs, but the total process is still complex. MODELPLEX tools such as EMFText and EuGENia reduce the complexity of working with the Eclipse environment.

Regarding how to apply MDE, we performed an experiment by developing two parallel solutions to the spread-context model. The evaluation was generally done by experts involved in the research. The work was done in iterations and with many trials and failures and is regarded as exploratory. The experiment showed that

- The DSL leaves the end-modeller with a simpler modelling language, thereby supporting a faster model development. Also, the writing of the code generators was simpler and faster with the DSL model. The complexity lies in using GMF in developing the DSL tooling.
- On the other hand, using a standard language such as UML or developing a UML profile has promises and pitfalls as well. There are several tools that support UML and UML profile development and developers are more familiar with UML tools. However, as pointed out by [5], the UML 2.0 profile mechanism does not provide a means for precisely defining semantics associated with extensions. For this reason, developers cannot use profiles in their current form to develop domain-specific UML variants that support the formal model manipulations required in an MDE environment.

Regarding the MDE approach, we found applying it very useful for simulation and testing purposes and have already started using it. We also performed modelling for architectural analysis. However, a full MDE approach, i.e. model-centric rather than model-assisted development, requires tools that specially support *iterative software development* with features such as evolvable metamodels and configuration management of models. Furthermore, in our view, the development approach should support what we call a *model-virtual-machine* (MVM), i.e. the execution and debugging has to happen at the model level. Today's practice with code generation does not hide the complexity of the underlying layers. The total complexity of system development therefore increases without an end-to-end MDE solution, especially when legacy code must co-exist with the models and the generated code. Until further technological advances allow the application of MDE throughout the entire development lifecycle, we will selectively apply it in the areas we found it useful.

6 Summarizing the experiences

In this section we combine lessons learnt in the three cases to identify areas where MDE is considered as beneficial and the main challenges.

6.1 Experiences regarding the MDE Approach

Industry partners in MODELPLEX have performed extensive evaluations of MDE in their contexts by using solutions developed in MODELPLEX as well as internal and external tools. Expert judgements, quantitative data on aspects such as performance, and interviewing developers and users have been the major sources of empirical data. The experiences of MODELPLEX partners show that the MDE approach was considered especially useful for:

(a) *Abstraction and hiding details* when a complex system of multiple systems is being developed, modelling is especially useful to break down the complex system into several layers or from different viewpoints. MDE techniques such as model composition and traceability between models are required to manage the complexity of developing complex and large models. Telefónica and WGO have taken advantage of abstraction techniques and composition tools in developing their complex systems.

(b) *Communicating with non-technical staff* developing domain-specific modelling languages or UML profiles improves communication with domain experts and business analysts by hiding non-relevant technical details. Several cases in MODELPLEX developed DSMLs or UML profiles for improving communication: in SAP with business process analysts, in Telefónica with domain experts in charge of telecommunication service specification and in WGO with software architects and security experts.

(c) *Simulation and model-based execution* a system must be analysed in the design phase in order to evaluate and improve the quality of its architecture, design or compositions, or to evaluate its performance. All of these cases need executable models and simulation facilities. In SAP, the MDPE workbench was developed for decision support for business experts. The MPDE workbench was customised by another industrial partner in the context of collaborative decision-making in the airport crisis management system. In WGO, hardware was simulated in order to test the system for various instrument failures. The validity of simulation results in both cases is however highly dependent on the quality of input data.

(d) *Model-based testing* in addition to the above three scenarios where MDE has proven to be useful, model-based testing has the potential to save effort and improve the testing process. WGO is already using model-based testing for black-box testing of a system under development by emulating the hardware. SAP and Telefónica also experimented with model-based testing and verification at the model level. More research is needed to develop mature tools in this area.

The companies in the project invested in developing DSLs that capture their domain concepts and transform these into code. The aspect of the MDE approach that deals with modelling Platform-Independent Models (PIMs) and then transforming these into Platform-Specific Models (PSMs) was not found to be relevant in the cases.

Our findings suggest that the participants in the research found MDE as useful for solving some problems of developing complex software systems while the methodology and tools are not generally perceived as easy to use. Both aspects however improve with more usage and companies with more experience on MDE such as SAP found it more useful and easy to use. The compatibility of MODELPLEX tools with one another was not considered as a problem while compatibility with external tools created some problems, for example for XMI import/export. One lesson learnt in the project was to develop solutions around industry accepted platforms such as Eclipse and adapt other tools to these environments.

6.2 Relation to the state of the art

In order to evaluate the state of the art regarding applying MDE in industry, we performed a systematic review in 2008 that covered papers published from 2000 to 2007 in some major journals and events related to MDE [14]. This review discovered 25 papers reporting experiences from different domains, although with very little quantitative data. Regarding motivations for applying MDE, the papers discussed:

(a) Increased productivity and shortened development time.
(b) Improved software quality by improving the quality of the generated code, improving the quality of intermediate models, earlier detection of bugs by model-based simulation and testing.
(c) Automation by generating code and other artefacts.
(d) Provision of a common framework for software development across the company and phases of the lifecycle.
(e) Maintenance and evolution concerns such as maintaining the architecture consistency from analysis to implementation and developing PIMs that have long life-span.
(f) Improved communication and information sharing among stakeholders.

The points on the top of the list are most frequently mentioned in the papers. In the MODELPLEX research it was not possible to collect enough quantitative data to verify benefits regarding productivity or quality improvements. Small-scale experiments would not generate valid data and observing these benefits would have required long-term follow up. The observed benefits in the industrial cases are related to model-based simulation and testing, improving maintenance and evolution by abstraction, and improved communication. However, these benefits are related to productivity and software quality. For example, late detection of performance issues increases the cost of fixing while possibly decreasing the quality of software due to workarounds. Model-based testing and simulation improves the quality of software and saves effort by detecting defects earlier in the development process.

In another MDE adoption study, the costs and benefits of MDE in the car industry were studied [11]. In this study, 12 interviews were conducted with car producers, their suppliers and consulting companies. As the motivation to use MDE, the following were mentioned: cost savings in the development, the fear of missing the next generation development technology, competitive pressure, and the experience that many innovative functions can only be developed via model-based technologies. The companies hoped to achieve the following advantages because of MDE: higher degree of automation, possibility of detecting errors earlier, better communication with colleagues and reduced effort for reusing functions in other car lines. Negative experiences were reported to be high dependence on tool vendors, bugs in the tools and the need for process redesign. Some of the interviewees also suggested that they cannot trust code generators to produce safety-critical and efficient code with the same quality as hand-written code, while others did not agree. Some of the advantages listed above are in-line with what we have observed in the project, especially regarding earlier detection of errors and better communication. The fear of missing new technology and hoping that MDE will solve some of the challenges in developing complex software systems have been motivations for participating in the research.

Single studies have also been published since 2007, which report on the motivations and benefits of applying the MDE paradigm, especially in the series of workshops on domain engineering.[13] Others have applied formal modelling techniques, such as in Ferrari et al. [4], who have applied formal model-based development using Simulink/Stateflow-based development in railway signalling. They estimate that a developer spends 30% more time on modelling than on coding. Nevertheless, this greater effort is balanced by the fact that notable cost reductions are achieved in terms of verification activities (with a time reduction of about 70%). We

[13] See http://domainengineering.org/ and http://www.dsmforum.org/.

did not collect data on effort to confirm this assumption but one motivation for applying model-based testing in MODELPLEX has been to save effort by generating test cases and reusing them. Ferrari et al. considered that their approach still needs improvement to ensure consistency of models at different abstraction layers which is also regarded as a challenge in MODELPLEX and is partly addressed by implementing traceability in transformations.

Experiences on applying MDE in the domain of industrial processes are discussed in [9], which describes the development of a UML profile for this domain. The authors note that the overhead of developing and customizing the transformation is worthwhile, especially in large projects, which can benefit from automating routine work. There is a great amount of reuse in this domain where thousands of objects are listed to develop control diagrams. They also report drawbacks, for example compatibility between different modelling viewpoints and models at different granularity levels. Also, the authors warn against putting too much effort on platform-independent modelling in this domain. The biggest challenge is the adequacy of modelling concepts, which shows that the domain models may change, as in the Telefónica case. The authors, however, emphasize that a possible advantage of MDE is integrating it with model-based simulation, which was also a conclusion of our cases.

6.3 Challenges

In the road toward project objectives, numerous challenges were identified:

- *Coping with changes in metamodels and standards*, both for UML and domain models. Current tools and development environments cannot manage the evolution of DSLs, profiles or models.
- *Getting the tools to interact properly* with one another and anything outside the domain, having a different underlying syntax to that of other languages.
- *Managing all the artefacts*, where models (from different viewpoints and at different abstraction levels) and transformations are added as artefacts to the software development process and the complexity of managing artefacts has increased considerably. Ideally, it should be enough to operate at the level of models in an MDE approach. Since in practice this is not yet possible, the complexity of software development in this aspect has increased. Taking into account that everything evolves as well, the migration of models is a concern. Tools for managing models and metamodels and traceability are of great interest.
- *Developing an MDE environment* requires high skill and expertise in domain concepts, language engineering, tool development, standards and transformations. Tools that facilitate this transition, such as domain ontologies and DSL development frameworks, are of value.
- The participants in this study are also concerned with the *scalability of the solutions and applying them to large-scale projects* where hundreds of developers are involved and it should be possible to perform iterative development. Effective model management tools are required to support the MDE approach in the development lifecycle of large-scale projects.

Some of the aforementioned challenges are introduced by the MDE approach and require suitable solutions. Models, metamodels and transformations are artefacts that should be managed, evolved and reused. Model semantics and managing relations between models are also important challenges in MDE. Other challenges are tool related such as incomplete implementations of standards, poor usability and the lack of documentation.

We would also like to emphasize challenges regarding collecting empirical data in research projects. The evaluation method here combined quantitative with qualitative assessment and observations with perceptions. While we hoped to collect more quantitative data, the nature of iterative development and continuous improvement of tools did not allow for the collection of meaningful data on effort spent on, for example, modelling or customization of tools. Comparing MDE with other development methods requires reliable baseline data, which is often missing. We also applied the Technology Acceptance Model [2] to summarize the developers' perceptions regarding MDE and used a questionnaire to collect opinions regarding specific tools and techniques. The detailed results of this study are going to be published in a separate paper while some results are presented in Sect. 6.1.

An important area for future empirical research is to evaluate the cost-effectiveness and Return-On-Investment (ROI) of MDE by performing more longitudinal case studies. ROI in our experience is related to the reuse of domain models, transformations, other artefacts and tools in several projects.

7 Final conclusions

Model-driven engineering has not been an out-of-the-box solution for any of the cases discussed in this paper. Applying the approach required developing metamodels and tools for the domain, customizing generic tools and integrating tools into a tool chain. This fact is both the strength and the weakness of the MDE approach: developing the environment requires high expertise and is costly, which does not pay off for small or single projects. However, once such a development environment is created, it has the potential to be easier to use than generic modelling tools and to save effort by automation. Although productivity improvements could not be measured in our cases, the potential exists based on the feedback from developers.

The cases presented in this paper describe how MDE was applied in three industrial cases using state-of-the-art tools. In these cases, MDE was evaluated to be especially useful for the purposes of simulation and execution, abstraction, communication and model-based testing and even more so given the possibility to reuse solutions in multiple projects.

The project identified several challenges and open issues. Better support for DSL development is a must before domain-specific solutions are widely applied. Tools must be improved regarding usability, multi-user support, versioning of models, conflict detection and resolution and diff/merge possibilities. All of these features are required for developing complex software systems.

In our opinion, the main question that an organization has to ask itself is "Where do I need MDE?" Companies may be reluctant to change their development processes and tools. However, if specific areas are identified where MDE can help and the cost of the transition can be justified by reducing risks or by reuse of solutions in multiple projects, the transition may happen gradually.

Acknowledgments This work was supported by MODELPLEX project (IST-FP6-2006 Contract No. 34081), co-funded by the European Commission as part of the sixth Framework Program (FP6). Alin Stefanescu was also supported by the Romanian research grant CNCS-UEFISCDI no. 7/05.08.2010.

Appendix

In this appendix, tools developed or extended within the MODELPLEX project are briefly presented. Tools in each technological area are listed in alphabetic order. Some of the tools do not have a website but contact information may be found on the MODELPLEX website (see tools section there).

Model engineering

ATL: ATLAS Transformation Language

(http://www.eclipse.org/atl/) ATL is a Model-to-Model transformation language and toolkit initiated from research in INRIA. It is widely adopted and is part of M2M Eclipse project. It was extended in MODELPLEX with traceability support.

AMW: ATLAS Model Weaver

(http://www.eclipse.org/gmt/amw/) AMW is a tool for establishing relationships (i.e. links) between models and is part of the same environment as ATL. The links are stored in a model, called weaving model, conforming to a weaving metamodel. These links may be used, e.g. in metamodel comparison, traceability, and model matching.

DSL evaluation framework

(http://quality-mde.org/) It is a framework to evaluate the quality of DSLs from different viewpoints, developed by SINTEF as part of research on quality of models in MDE. Publications are available on the website. Several DSLs were developed in MODELPLEX and we used the framework to evaluate a few of them.

Epsilon environment and languages

(http://www.eclipse.org/gmt/epsilon/) Epsilon is a family of languages which may be used to interact with EMF models to perform common MDE tasks. The environment is developed by York University and contains several tools such as ECL—Epsilon Comparison Language, EML—Epsilon Merging Language and EVL—Epsilon Validation language, all available as Eclipse plug-ins and free to download.

EMFText

(http://www.emftext.org/) EMFText is a tool for defining a textual syntax for modelling languages, for example textual representations of EMF models. It can be used together with Reuseware on textual languages. It is developed by Dresden University of Technology as an Eclipse plug-in and it is free to download.

EuGENia tool for developing GMF editors

(http://www.eclipse.org/gmt/epsilon/doc/articles/eugenia-gmf-tutorial/) Implementing a visual editor using the built-in GMF facilities is a particularly complex and error-prone task. EuGENia is a tool that automatically generates the .gmfgraph, .gmftool and .gmfmap models needed to implement a GMF editor from a single annotated Ecore metamodel. The tool is part of the Epsilon environment.

MoDisco: model discovery

(http://www.eclipse.org/MoDisco/) MoDisco is an Eclipse GMT component and is composed of a set of model-driven reverse engineering tools, named discoverers. INRIA and SODIFRANCE were partners in MODELPLEX who developed the platform.

MOFScript

(http://www.eclipse.org/gmt/mofscript/) MOFScript is a tool for model-to-text transformations, e.g. to support generation of implementation code or documentation from models. The

MOFScript tool is based on EMF and Ecore. The tool was developed within the MODELWARE project by SINTEF. It was later extended in MODELPLEX to achieve compliance to OMG MOF Model to Text (M2T) standard and traceability support during transformations. The traces can be analysed by the TraMDE tool. It as an Eclipse plug-in and it is free to download.

MontiCore

(http://www.monticore.de/) The tool can be used for developing compositional textual domain specific languages. Languages and accompanying tools can be developed separately and combined at runtime. During this combination, the semantics and the functionality of the partial languages and tools are preserved. MontiCore is available as an OSTP (Online Software Transformation Platform) service while the Eclipse platform is chosen as a target for editor generation. It is developed by RWTH Aachen University.

Reuseware composition framework

(http://www.reuseware.org/index.php/Reuseware) Reuseware is a framework for creating composition systems for existing or newly developed modelling languages. The users are able to compose model elements, compose diagrams, define reusable facets and even modify composed models and reflect back changes to the fragments. The tool was used in MODELPLEX with Eclipse GMF, TOPCASED and Rational Software Architect (RSA), as well as with textual modelling languages. It is developed by Dresden University of Technology as an Eclipse plug-in and it is free to download.

System Grokker

(https://www.research.ibm.com/haifa/projects/services/grokking/) The tool is a model-based software architect assistance technology developed by IBM Haifa Research Lab to support the incremental and iterative user-driven understanding, validation, and evolution of complex software systems through higher levels of abstraction.

TraMDE: trace analysis tool

(http://www.modelbased.net/modelplex/traceability/) TraMDE Trace Analyser is a tool capable of storing trace information as a model. Traces can be established both manually by the user from tools like Papyrus UML and GMF-based editors and automatically by tools like the MOFScript Model-to-Text transformation engine. In addition, different views and analysis functionalities, such as simple impact analysis, are supported. It is developed by SINTEF as an Eclipse plug-in and it is free to download.

Model-based verification and validation

As part of developing a Simulation, Validation and Testing (SV&T) Workbench in MODELPLEX, several tools were developed, in addition to a Testing Metamodel. Tools are listed here and we refer to the project website for more information.

BCC: behavioural consistency checker

(http://move.lip6.fr/software/BCC/) BCC is a model-checking tool for checking the consistency of UML diagrams. It was developed by LIP6 as an Eclipse plug-in. The prototype version can be downloaded for free.

EVL: Epsilon Validation Language

(http://www.eclipse.org/gmt/epsilon/doc/evl/) EVL is a validation language developed by York University built on top of Epsilon Object language for defining and checking constraints on models. EVL constraints are quite similar to OCL constraints. However, EVL also supports dependencies between constraints (e.g. if constraint A fails, do not evaluate constraint B), customizable error messages to be displayed to the user and the specification of fixes, which users can invoke to repair inconsistencies. It is developed as an Eclipse plug-in and it is part of the Epsilon framework.

Escalator tool for refinement verification

(http://heim.ifi.uio.no/~massl/escalator/) Escalator is a tool for verifying the refinement of UML2 sequence diagram specifications, based on the formal definition of refinement. It can be used in combination with any UML tool for specifying sequence diagrams in EMF-based XMI. It is developed by SINTEF as an Eclipse plug-in.

FOKUS!MBT: test derivator tool by Fraunhofer FOKUS

FOKUS!MBT represents an UML based test generator that supports the derivation of test suites from UML system models augmented with the U2TP (UML2 Testing Profile). The tool has a proprietary license. For contact details, please refer to MODELPLEX website.

IBM Model Debugger and test generator

IBM Model Debugger is a tool for the execution of behavioural UML models and the visualization of the execution by animating the behavioural diagrams. The Test Generator automatically generates tests based on a behavioural UML model. For contact details please refer to MODELPLEX website.

Metrino

(http://www.modelbus.org/modelbus/index.php/metrino) Metrino is an integrated set of tools to support the validation and quality assurance of models based on OCL (Object Constraint Language) and SMM (Structured Metrics Metamodel). Models can be developed in UML or a DSL based on MOF. It is developed by Fraunhofer FOKUS as an Eclipse plug-in and it is free to download.

MBTM: Model-Based Testing Metrics

The MBTM tool allows measuring test quality in a language independent way. It is an open source tool developed by RWTH Aachen University. For contact details please refer to MODELPLEX website.

Simulation

MDPE: Model-Driven Performance Engineering

(http://reuseware.org/index.php/MDPE/) It is a workbench developed in MODELPLEX by Dresden University of Technology, SAP Research and XJ Technologies for analysing business processes from a performance perspective by using historical data. More detail is provided in Sect. 3.2 of this paper.

AnyLogic

(http://www.xjtek.com/) AnyLogic is a commercial simulation tool developed by XJ Technologies. The company was a MODELPLEX partner.

Other

AM3: AtlanMod MegaModel Management

(http://www.eclipse.org/gmt/am3/) The goal of AM3 is to provide an integrated environment for the management of the various modelling artefacts coming from a MDE development process. It is developed by INRIA.

Lightweight repository and team provider

This tool is a model-based repository solution for the combined management of models and other artefacts. The client side uses an Eclipse plug-in that implements the Team Provider Interface. In the version available in MODELPLEX, the granularity was at the model level and not at that of model fragments. It is developed by Fraunhofer FOKUS and has an EPL (Eclipse Public License) license. For contact details please refer to the MODELPLEX website.

References

1. Brown, A.: An Introduction to Model Driven Architecture—Part I: MDA and today's systems. The Rational Edge (2004)
2. Davis, F.: Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Q **13**(3), 318–339 (1989)
3. Evans, A., Fernández, M.A., Mohagheghi, P.: Experiences of developing a network modeling tool using the Eclipse environment. In: Proc. of ECMDA-FA'09. LNCS, vol. 5562, pp. 301–312. Springer, Berlin (2009)
4. Ferrari, A., Fantechi, A., Papini, M., Grasso, D.: An industrial application of formal model-based development: the Metro Rio ATP case. In: Proc. of 2nd Int. Workshop on Software Engineering for Resilient Systems (SERENE 2010). http://www.dsi.unifi.it/~fantechi/Collaborazionegets/pubblicazioni.html
5. France, R.B., Ghosh, S., Dinh-Trong, T., Solberg, A.: Model-driven development using UML 2.0: promises and pitfalls. IEEE Comput. **39**(2), 59–66 (2006)
6. Fritzsche, M., Gilani, W., Fritzsche, C., Spence, I.T.A., Kilpatrick, P., Brown, T.J.: Towards utilizing model-driven engineering of composite applications for business performance analysis. In: Proc. of ECMDA-FA'08. LNCS, vol. 5095, pp. 369–380. Springer, Berlin (2008)
7. Fritzsche, M., Picht, M., Gilani, W., Spence, I., Brown, J., Kilpatrick, P.: Extending BPM environments of your choice with performance related decision support. In: Proc. of Business Process Management (BPM'09). LNCS, vol. 5701, pp. 97–112. Springer, Berlin (2009)
8. Fritzsche, M., Gilani W., Lämmel R., Jouault, F.: Model transformation chains in model-driven performance engineering: experiences and future research needs. In: Proc. of Modellierung 2010. LNI, vol. 161, pp. 213–220. Springer, Berlin (2010)
9. Hästbacka, D., Vepsäläinen, T., Kuikka, S.: Model-driven development of industrial process control applications. J. Syst. Softw. (2011, accepted) doi:10.1016/j.jss.2011.01.63
10. Johannes, J., Fernández, M.A.: Adding abstraction and reuse to a network modelling tool using the Reuseware composition framework. In: Proc. of ECMFA'10. LNCS, vol. 6138, pp. 132–143. Springer, Berlin (2010)
11. Kirstan, S., Zimmermann, J.: Evaluating costs and benefits of model-based development of embedded software systems in the car industry—results of a qualitative case study. In: Proc. of 5th C2M:EEMDD Workshop at ECMFA'10, pp. 18–29 (2010)
12. Kelly, S., Pohjonen, R.: Worst practices for domain-specific modelling. IEEE Softw. **26**(4), 22–29 (2009)
13. Kätker, S., Patig, S.: Model-driven development of service-oriented business application systems. In: Business Services: Konzepte, Technologien, Anwendungen. Wirtschaftsinformatik, Band 1, pp. 171–180. Österreichische Computer Gesellschaft (2009)
14. Mohagheghi, P., Dehlen, V.: Where is the proof? A review of experiences from applying MDE in industry. In: Proc. of ECMDA-FA'08, LNCS, vol. 5095, pp. 432–443. Springer, Berlin (2008)
15. Mohagheghi, P.: An approach for empirical evaluation of model-driven engineering in multiple dimensions. In: Proc. of 5th C2M:EEMDD Workshop at ECMFA'10, pp. 6–17 (2010). http://www.esi.es/modelplex/c2m/papers.php
16. Schmidt, M., Polowinski, J., Johannes, J., Fernández, M.A.: An integrated facet-based library for arbitrary software components. In: Proc. of ECMFA'10, LNCS, vol. 6138, pp. 261–276. Springer, Berlin (2010)
17. Stefanescu, A., Wieczorek, S., Kirshin, A.: MBT4Chor: a model-based testing approach for service choreographies. In: Proc. of ECMDA-FA'09. LNCS, vol. 5562, pp. 313–324. Springer, Berlin (2009)

18. van Amstel, M., Lange, C., van den Brand M.: Metrics for analysing the quality of model transformations. In Proceedings of the 12th ECOOP Workshop on Quantitative Approaches on Object Oriented Software Engineering, pp. 41–51 (2008)
19. Wadler, P.: Deforestation: transforming programs to eliminate trees. In: Proc. of the 2nd European Symposium on Programming, pp. 231–248. North-Holland, Amsterdam (1988)

Author Biographies

Parastoo Mohagheghi is a research scientist at SINTEF and Adjunct Associate Professor at The Norwegian University of Science and Technology (NTNU). She received her Ph.D. in Information and Communication Technology from NTNU in 2004. She has participated in several national and international research projects and has industry experience from Ericsson in Norway. Her research interests include software quality, model-driven development, software reuse, service engineering, cloud computing and empirical studies. She is a member of IEEE.

Wasif Gilani holds a Ph.D. in Computer Science from Friedrich Alexander Universitaet Erlangen-Nuernberg Germany and a Masters degree in Computational Engineering from the same university. Wasif has work experience in different industries like oil and gas, manufacturing and IT. Since 2007 he has been working for SAP Research and has been leading the research areas of process-oriented business process performance management, business continuity management and model-driven engineering. His research interests also include service engineering, software product line engineering and adaptable systems.

Alin Stefanescu is a researcher at the University of Pitesti in Romania. He received his Ph.D. in computer science from University of Stuttgart and his M.Sc. from University of Bucharest. His research career path intersected both academic and industrial worlds. In academia, Alin investigated software validation and verification at European universities in Edinburgh, Munich, Bucharest and Konstanz. In industry, at SAP Research, he made contributions in model-based testing for service-oriented architectures and transferring the results into the SAP development groups.

Miguel A. Fernandez is currently a project manager at Ericsson. He worked in Telefónica I+D in Valladolid, Spain before joining Ericsson, where he participated in several research projects. He holds a computer science degree from Universidad de Oviedo. His research interests include model-driven engineering, domain-specific languages, business process modelling, autonomic computing and their application to the telecommunications industry and, in particular, to remote network management.

Bjørn Nordmoen has a Master of Physics from University in Oslo (1983) and has been working at WesternGeco in the R&E department since then. He has been the Chief System Architect at the Oslo Technology Center since 2001, where he overlooks and mentors software architecture activities. During the past years he has been involved in several EU projects such as MODELPLEX, ModelWare, COMBINE and OBOE.

Mathias Fritzsche is a developer for Modeling Methodologies and Taxonomy at SAP AG. He holds a Ph.D. in Computer Science from the Queens University Belfast and a M.Sc./B.Sc. in Software Systems Engineering from the Hasso Plattner Institute, Potsdam University. Mathias has a research interest in model-driven engineering, in particular methologies and tool architectures for modular creation of models. Additionally, he has a strong interest in employing models for process performance analytics, process optimizations and business continuity management.