

Automated Model Driven Development Processes

Aitor Aldazabal¹, Terry Baily¹, Felix Nanclares¹,
Andrey Sadovykh², Christian Hein³, Tom Ritter³

¹European Software Institute
Parque Tecnológico de Bizkaia #204
E-48170 Zamudio, Bizkaia-Spain
{aitor.aldazabal|terry.bailey|felix.nanclares}@esi.es

²Softeam
21 Avenue Victor Hugo
75008 PARIS, France
andrey.sadovykh@softeam.fr

³Fraunhofer Institute FOKUS
Kaiserin-Augusta-Allee 31
10589 Berlin, Germany
{christian.hein|tom.ritter}@fokus.fraunhofer.de

Abstract. Software developments of large scale systems require well organized process structures for a proper and smooth execution. A number of frameworks already exist for outlining and defining processes and their workflows and work break down structures. The different activities which compose these processes need to be executed in an organized fashion, each activity having work products as inputs and outputs. In that sense the individual activities can be considered as services which need to be executed. Considering the whole development life cycle of software development of complex systems there are lots of services which have to be called in an orchestrated way to successfully create a software system in the end. This paper describes a novel integration of a process definition framework, which is used to define and to enact development processes, an orchestration tool, which allows the definition and execution of service compositions, and ModelBus, a service oriented middleware connecting model-based development tools and the services they offer. This integration provides major benefits in particular for medium-sized and large-sized development teams. The paper describes the general architecture of the solution as well as the individual benefits.

Keywords. Process Enactment, Automation, Tool Integration, Modeling Services

1 Introduction

In a model-driven development process a lot of actions are performed on models, among them model transformation, model composition, model verification, model checking, model storage, or code generation. Typically, there are multiple different roles participating in a development process which have different interest in the individual parts of the systems. They use tools appropriate for accomplishing tasks part of a specific process step. Due to the variety of tasks and process steps the used tools differ very much in their functionality and user interface. For example a requirements management tool is to some extent different from a test execution engine. Thus both tools are used in the same development process but have different views on the system and are used by different users. This in particular makes it really challenging to form a chain of tool for building software systems which is highly flexible as well as seamlessly integrated and which has a high degree of automation. The individual tools of a tool landscape offer certain functionality and rely on certain preconditions with respect to the expected inputs. For example a model transformation tool expects input models of a specific type and realised with a specific modelling framework. There are other tools which are meant to be used by humans and offer a graphical user interface. The user interface visualizes functionality realised by a tool or even provided by other tools. For structuring the discussion about tool integration we identify based on [1] the following four layers:

- **Data Integration.** To integrate different tools which act on data (in this case act on models) they need a common understanding what kind of data they handle and how the data is encoded. It is needed to identify types of models (e.g. UML 2.1 model [6]) and the modelling framework (e.g. MOF[7]) and/or serialisation used.
- **Control Integration.** If tools offer certain functionality which can be used by other tools it is vital to have a clear description of the service offered and of the way how to use this functionality. So the access mechanisms or transport protocols (e.g. Web Services using SOAP over HTTP) need to be identified.
- **Process Integration.** To form a development process it is needed to define process phases and steps in which different activities get accomplished. The execution of the process relies on functionality provided by individual tools as well interaction with human beings. Process Integration is targeted on how to orchestrate functionality, roles and data flows in a global process perspective.
- **Presentation Integration.** Another level of integration arises from the need of an integrated user experience. So a user want to use the best possible presentation of data based on his specific context and role he is acting in. There is always the difficulty to present information in the appropriate format and granularity.

Taking a look onto this problem and in particular by considering Eclipse as an integration platform one way to create a basic integration would be to write or to use Eclipse Plug-ins for all the needed functionality and then load all of them into a single Eclipse workspace. In this approach even a common perspective could be created, which would serve the purpose of presentation integration. Obviously this is not a great "integration", it's more like a common packaging but if the tools do not use each

other's services then, it is hardly possible to really call this integration since all the other aspects of integration are not considered sufficiently.

MDE and MDD processes involve multiple heterogynous model-based tools to be used in so-called “tool chains”. Indeed, working on large software projects will impose using code and model sharing solutions – CVS, SVN model repositories; Modellers able to show different type of systems views – goal and requirements models (BPDM), business models (BPMN [13]), conceptual, architectural and design models (UML2 with different types of profiles, DSLs); Model Transformation and Code Generation tools; Verification, Simulation and Testing tools. Developers use several of them at each step during the whole project life-cycle. Commonly, developers execute “modelling services” manually moving models from one tool to another and specifying execution parameters.

Efficiency of this manual process can be improved, when some common process patterns are discovered or imposed by software engineering process. In this case, these common processes can be automated provided that tools implements ModelBus interfaces. For example, for the standard quality procedures it may be required to run a set of standard quality checks – naming conventions, requirements description coverage, documentation coverage. Depending on model size the process may be long and hard. Automating it would allow save time and provide more control to the software quality.

This paper presents a solution to the first three levels of integration needs; namely data, control and process integration. Additionally, it discusses possible solutions for the presentation integration. The solution is created as part of two European projects co-funded by the European Commission. The first one is Modelware [8] which has been finished already and the second one is Modelplex [9] which is currently running. In particular these projects evaluate the solution presented herein based on industrial case studies. The remainder of the paper is organized as follows. Section 2 describes the basic infrastructure we used for tool integration, which is the ModelBus. Section 3 explains how automation can be achieved by creating and executing service orchestrations. Section 4 explains how this is embedded in the definition and execution of development processes. Section 5 concludes the paper.

2 A world of services

Simply putting a set of Eclipse Plug-ins into one Eclipse instance will give a unified graphical interface to a set of tools but is mainly about presentation integration. It does not cover the other three aspects (as discussed above). Now improving on this idea, something like the Eclipse Team Provider interface could be created. This is, provide a common interface (as API not GUI) which allows to group similar services provided by different providers. In the Team Server/Provider case we can usually see the case where several CVS and SVN clients can be installed in the same workspace. However here we would be achieving a common entry point for similar application but there is no real integration between the applications because they do not cooperate in any standard way with each other.

4 **Aitor Aldazabal, Terry Baily, Felix Nanclares,
Andrey Sadovkyh, Christian Hein, Tom Ritter**

Putting this Eclipse Plug-in based approach another step forwards would be to use or contribute to the Eclipse Communication Framework (ECF) which aims at providing a set of common interfaces for applications to extend and use to communicate to each other even in a remote fashion. However, this framework is targeted on Eclipse based tools only and does not really address the specifics arising from dealing with models. However, staying only in the Eclipse World is in some case not feasible. There are many tools used in today's development processes which are not Eclipse based but have particular functionality and user interface. Engineers are trained to use those tools and these tools cannot be easily replaced.

For this reason we have designed and realized an Open Architecture for integration of arbitrary software tools (including those which are Eclipse based) using well established and open standards. This is done by following a service oriented approach. So we have a distributed platform where tools provide and consume services in order to obtain some added value. This is actually the definition of a SOA. We call this platform ModelBus. One could think of ModelBus as some sort of middleware, like CORBA [10], which has a registry with the services offered by each tool and allows you to consume services without worrying about the distributed nature of the system.

Using plain Web Services is another option to create a SOA for integrating software tools, actually ModelBus is compliant to some point with it. REST[11], XML-RPC[12] or any other protocols and architectures can prove to be useful but they do not care about the content of the data they transport. And this is exactly the key benefit of ModelBus.

Finally, using ModelBus, Process Enactment and Process Orchestration tools can ultimately be used to create/orchestrate/monitor composite services by combining the different services from the different tools into a workflow. And these tools are described later in the paper. Therefore the end user interface could be the workflow in a high level, business oriented notation, with a button which says execute and all the tools and services will start to run, windows will pop-up to each involved user and everything can be "automated" to a certain extent.

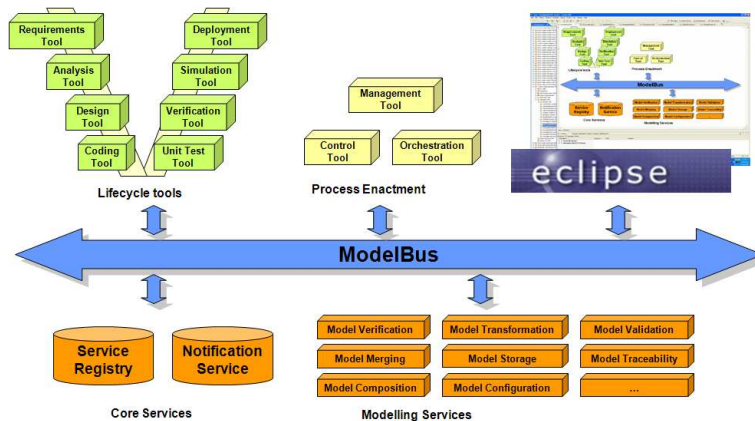


Fig. 1. ModelBus General Architecture

ModelBus is available as Open Source Software as part of the MDDi Project [4] hosted by Eclipse. Details on the ModelBus architecture are described in [2, 3] The following will shortly outline the key concepts. In ModelBus a tool can consume or provide services. These services are called modeling services. A modeling service can offer arbitrary functionality, for example the transformation of an input model into an output model based on a specific rule set. ModelBus uses the concept of adapter to bridge different technologies. Whenever a tool exposes a modeling service it will be deployed to ModelBus by registering its adapter in the ModelBus Service Registry. If a tool wants to consume a modeling service it simply calls the service via its adapter. The adapter makes a service lookup by using the Service Registry and forwards the actual service invocation to a tool offering the service. The service invocation can be made either locally or remotely in case a tool offering the requested service is only available remotely. But this is transparent to involved tools. Beside the tools used within the different phases of the development lifecycle there are some particular ones, which are targeted towards the automated enactment of the development process, which is outlined in the next sections.

3 Orchestration

Common platform for modelling services, ModelBus, helps to bring automation to the software engineering process. The Orchestration Tool developed in the frame of ModelWare/ModelPlex projects is a tool for automating execution of ModelBus services - modeling services provided by MDA tools. The orchestrations leverage on BPEL – Business Process Execution Language – [14] a language for orchestrating web services. This language provides a flexible notation for describing process – different types of service invocations, transitions between invocations, conditional transitions, loops, event handlers and rollback/compensation activities. The Orchestration Tool brings this flexibility to the modeling services domain. In addition, using BPEL allows using ModelBus services and conventional Web-Services in the same process. In particular, this allows managing human interactions involved in the process either by integrating custom Web Services like mail notifications either by integrating modeling services in BPEL4People or WS-HumanInteractions processes.

In order to illustrate the above-provided description let us consider a quality checking example in details. The following Eclipse BPEL diagram provides details on the process.

- 6 **Aitor Aldazabal, Terry Baily, Felix Nanclares, Andrey Sadovykh, Christian Hein, Tom Ritter**

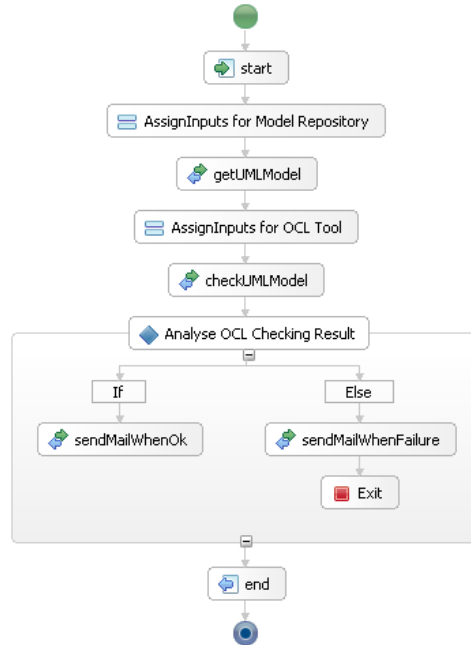


Fig. 2 Quality Constraints Checking BPEL Orchestration

The process involves Model Repository and OCL Checking Engine. At the first step, the tool retrieves a required model from the repository. Then, the verification is started to the given model and set of quality constraints. Depending on the result report on success of failure is send to involved developers.

This process involves both ModelBus services – model repository and OCL [16] verification – and conventional Web Services for interacting with humans – mail notifications. One may argue for integrating of these functionalities directly to the development environments or checking the constraints in run-time during the model composition. Nevertheless, depending on the size of the model and complexity of the verification constraints the process can require to be applied to the final results of the modeling and may take a long time blocking the workstations. In addition, using BPEL engines allows ensuring process execution persistency.

4 Process Enactment

4.1 Overview

Service Orchestration is used to coordination to invocation of modelling service to complete specific activities. The next step is to place these activities into a software engineering process taking the project management into account. Traditional project

management and planning tools allow project leaders to work in a free, manual, programming like environment which leaves to the planner most of the responsibility. These environments usually force the project leader to repeat tedious tasks while planning, monitoring and executing the project. Tasks which, by themselves, do not provide any added value to the final products, but instead waste some serious amounts of time during the planning and execution of any project, thus increasing the overall costs.

Sometimes these problems are palliated by using templates, should the tools support these. However in large organizations, where complex projects usually take place, each of the processes which compose the life cycle of a project can be viewed from several levels of abstraction. Also, from many areas of the organization several quality and standardisation requirements can be required in any of these life cycle processes, therefore rendering these solutions (the templates) inefficient.

In this section the idea of these problems being addressed by using MDE techniques, is proposed, techniques which are usually found in Software Engineering domains. The basis for this is the assumption that the traditional planning and monitoring tools resemble traditional programming languages. Meaning that traditional planning tools usually provide a single low level viewpoint of the project which ends up presenting the same problems as it does in traditional software coding.

MDE methods have proven their usefulness in software programming areas and, even if the business oriented project planning and monitoring domain differs from the more technical software development domain, the nature of the problems which arise in both is similar and hence, the solutions may be as well similar.

In order to apply MDE methods in project management, means to model the processes and entities which form up the domain are needed. In this matter there are already several meta-process modelling initiatives like Software Process Engineering Meta-model (SPEM) [15] and Business Process Modelling notation (BPMN) [13], both are initiatives from the Object Management Group (OMG). These modelling notations can be considered as the equivalent of UML for the business process domain.

By having the possibility of representing the information of the project planning's, processes, work products and all those elements which end up conforming the lifecycle of a project in a unified structured fashion, the possibility of applying all kinds of transformations to these models becomes available. These transformations will allow seamlessly creating several viewpoints or levels of abstraction, integrating information from various sources automatically, inferring new information and ensuring that the quality requirements are met by using existing model validation techniques.

The Software Process Engineering Meta-model, SPEM aims at providing organizations with the means to define a conceptual framework, which can provide the necessary concepts for modeling, interchanging, documenting, managing and presenting their development methods and processes. It constitutes a sort of "ontology" of software development process. It is the model underlying the Rational Unified Process, and to a lesser extent many of the IBM Global Services Methods. It serves as the basis for the construction of process authoring or management tools, such as IBM's RPW, Osellus's IRIS, Softeam's Objecteering. It also serves as the basis for further developments in project planning and estimation, resources

**8 Aitor Aldazabal, Terry Baily, Felix Nanclares,
Andrey Sadovykh, Christian Hein, Tom Ritter**

management, process simulation, etc. The actual enactment of processes—that is, planning and executing a project using a process described with SPEM, is not within its scope.

The Business Process Modeling Notation (BPMN) is a standardized graphical notation for drawing business processes in a workflow. BPMN was developed by Business Process Management Initiative (BPMI), and is now being maintained by the Object Management Group since the two organizations merged in 2005. Its current adopted version is 1.0 and the proposed one is 2.0. The primary goal of BPMN is to provide a standard notation that is readily understandable by all business stakeholders. These business stakeholders include the business analysts who create and refine the processes, the technical developers responsible for implementing the processes, and the business managers who monitor and manage the processes. Consequently BPMN is intended to serve as common language to bridge the communication gap that frequently occurs between business process design and implementation.

Both SPEM and BPMN relate in the fact that both can be used to describe workflows of processes and tasks pertaining to any project, product or activity. It can be observed that BPMN focuses more on the actual workflow of the activities, the relation between each artefact and their association with each activity, which is comparable to an UML activity diagram. SPEM on the other hand does not allow such a fine workflow representation but has a much more rich ontology which allows for representing aspects of the processes that cannot be represented in BPMN such as roles, decomposition and states of the artefacts, guidelines and work breakdown structures.

SPEM 2.0 is mainly used to define software processes that interested organisations want to describe. There exist several reasons for which organisations need to define their processes. These reasons go from quality models compliancy such as CMMI® to better know how the organisation performs its business by explicitly describing their processes. In fact using SPEM 2.0, they only specify their software processes at global level.

But SPEM, although specifically conceived for software development, maintains for most part of the specification a high enough abstraction level which may allow using it in other vertical domains such as banking, automotive, etc.

However this is not enough for organisations, as they need to enact their processes, meaning that they need to obtain specific, detailed and executable plans and they require mechanisms to monitor and control them, ensuring compliancy with standards.

Therefore, on their own SPEM and BPMN provide insufficient tools for enterprises to manage their projects and products, but combining the two greatly improves their potential since each of them, although they overlap a lot, have their main strengths focused in different areas, becoming a natural complement to each other.

BPMN will prove especially useful for the automation of the execution of tasks in the project since there are studies, mappings and tooling support to transform BPMN model to BPEL models, which can be executed inside a SOA context, which is in our case the ModelBus tool integration layer.

4.2 Fusing Enactment with orchestration.

During the MODELPLEX Project the Eclipse Process Framework Composer was modified to be used as the core tool to model SPEM processes. The vanilla tool was modified to enable model transformations from SPEM models to BPEL and MSProject. The embedded transformation tool is Open Architecture Ware and we perform 2 types of transformations, model to text to generate the BPEL and MSProject files and one model to model transformation which generated Enactment Models from SPEM models. In figure 2 it can be observed how these elements interact with each other.

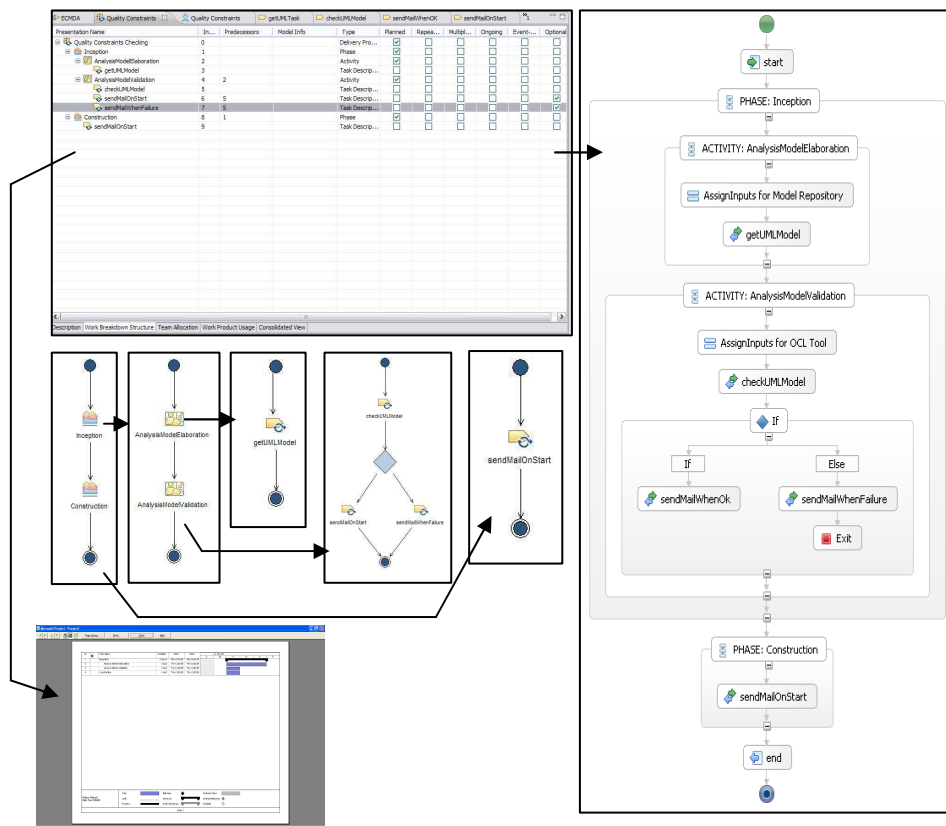


Fig. 2. Process Enactment for Quality Constraints Checking

4.3 Example Use Case of MDE Techniques for Process Enactment

In order to illustrate the coupling of the software engineering process with orchestrations of MDD tool chains, let us consider the following oversimplified example.

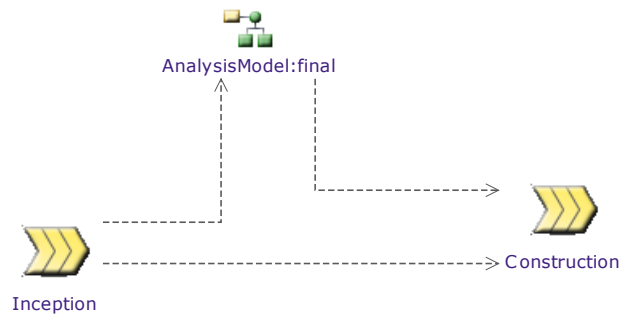


Fig. 3. Simplified Software Engineering Process

A RUP-like process consists of two phases Inception and Construction. During the Inception phase an Analysis Model is elaborated, validated and released. It is then transferred to the development team which implements Construction phase.

Based on this process, a project plan is created, which defines number of iterations, dates, human resources allocated and etc. Starting from this and using iterative transformations from SPEM to BPMN and then BPEL, an executable process can be defined [5]. In this process the interactions with involved teams and orchestrated process patterns is defined. Figure 4 depicts a final result for an engineering process using an orchestration defined in the previous section.

This is a representation of the project planning in an executable form. The process can be run with BPEL engines and can be monitored using available interfaces. This gives an opportunity to trace the execution of the project and synchronize the actual project planning with its executable representation.

In this way, the approach allows integration of heterogeneous MDD tools in orchestrated tool chains that can be coupled with project execution environments using process enactment techniques.

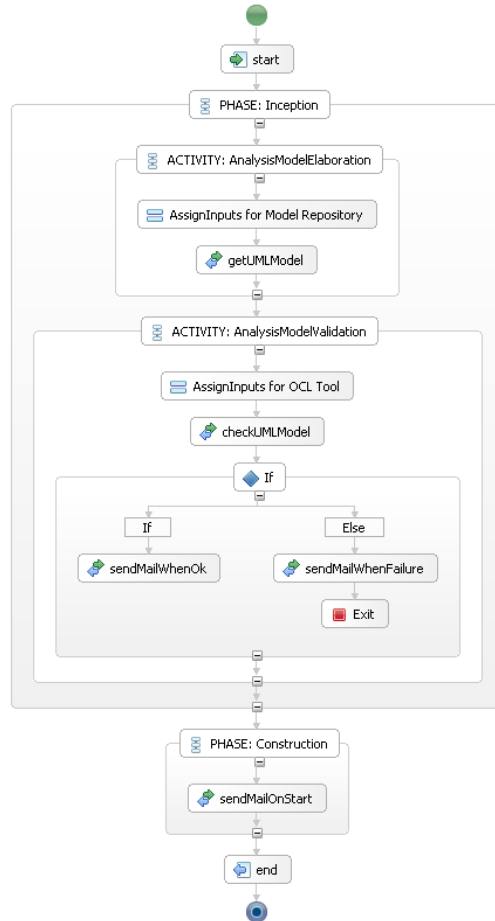


Fig. 4. Process Enactment for Quality Constraints Checking

5 Conclusion

In this paper we presented a general approach for creation of integrated tool chains with a link to process definition, process planning and process execution. This approach was followed by using a service oriented tool integration framework named ModelBus. It realises a SOA by allowing tools to provide and consume modelling service. Based on that, the service orchestration is in particular useful to automate service invocation. The orchestration in our case is achieved by using a BEPL engine capable of orchestrating modelling services and plain web service. Those service orchestrations build the fundamental building blocks in the development process definition of a model-driven development process definition and for its enactment. Those service orchestrations are used, when the actual process steps are executed. The

whole approach is instrumented by using standard technology like Web Service, BPEL, or SPEM.

There is still an open point with respect to the presentation integration and in particular to user interaction. One of the ideas which have been proposed to address this issue is to implement the user interactions as services themselves. Therefore one service implementation can invoke another service to request data. The implementation of the later service would pop up a window or requests the user for some kind of input. To address the presentation aspect of the integration a service oriented approach can be used. A service would be invoked by a client which would render the output to the user. We have already identified this issue but we have not solved all the problems. For example, in order to render the output of a model simulator the client or service requester may need some additional software to be installed on their terminal. This is not addressed by SOA architectures but the provisioning mechanisms of Eclipse would allow to remotely install and to provide the necessary bundles to client applications (Myinstall, P2). This will be addressed in future activities by integrating appropriate presentation integration solutions.

However, with the presented approach we can integrate heterogeneous tools forming a MDD tool chain. This comprises consuming and producing tools. We achieved the integration on 3 different layers (namely data, control, and process). We also integrate pragmatic approaches for the presentation integration, but we want further investigate this problem in upcoming activities.

One of the major benefits of our approach is the coupling of these integrated tool chains with the MDD software engineering process. This has the potential to substantially increase the productivity of today's software engineering. Obviously, it is a great support for automation but it gives also some post-execution capability to examine completed development processes, for example to analyse particular problems. The presented solution will be exploited in the Modelplex project. Within this project four industrial case studies will be executed, targeting on creation of complex systems.

References

- [1] I. Thomas, B.A. Najmeh. Definitions of tool integration for environments. IEEE Software, 9(2):29-35, March 1992
- [2] X. Blanc, M.-P. Gervais, P. Sriplakich, Model Bus. Towards the Interoperability of Modelling Tools, Proceeding of the European workshop on Model Driven Architecture: Foundations and Applications (MDAFA'2004), June 2004, Linköping University, Sweden, selected for : Lecture Notes in Computer Science (LNCS) « Model Driven Architecture: Revised Selected Papers », Volume 3599/2005, Springer
- [3] Prawee Sriplakich. ModelBus - An Open and Distributed Environment for Model Driven Engineering, Ph.D, September 2007
- [4] Model Driven Development integration – MDDi, www.eclipse.org/mddi
- [5] Reda Bendraou, Andrey Sadovykh, Mari-Pierre Gervais and Xavier Blanc. Software Process Modeling and Execution: The UML4SPM to WS-BPEL Approach. Proceedings of IEEE EUROMICRO 2007, pp. 314-322, ISBN 0-7695-2977-1
- [6] OMG Document - UML – Unified Modelling Language (UML) Specification: Infrastructure. <http://www.omg.org/docs/formal/07-02-03.pdf>

- [7] OMG Document – MOF – Meta Object Facility formal/02-04-03.pdf
- [8] ModelWare Project <http://www.modelware-ist.org/>
- [9] ModelPlex Project. <http://www.modelplex.org>
- [10] OMG Document – CORBA – Common Object Request Broker Architecture
http://www.omg.org/technology/documents/corba_spec_catalog.htm
- [11] Representational State Transfer <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>
- [12] Extensible Markup Language Remote Procedure Call -XML-RPC Specification
<http://www.xmlrpc.com/spec>
- [13] OMG Document - Business Process Modeling Notation <http://www.omg.org/cgi-bin/apps/doc?dtc/06-02-01.pdf>
- [14] OASIS Document - Business Process Execution Language <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>
- [15] OMG Document - Software Process Engineering Metamodel
<http://www.omg.org/technology/documents/formal/spem.htm>
- [16] OMG Document – OCL - Object Constraint Language Specification Version 2.0.
<http://www.omg.org/docs/ptc/05-06-06.pdf>