Abstract

This paper describes the automation and optimization of the software development process of NTT Data. The development process defines the business modeling, requirement capturing as well as, system planning and implementation activities to be carried out by the development project members. The target systems of this development process are large web based information systems implemented by leveraging Java/Java EE 5 technologies. The automation and optimization of that process was achieved by the application of model driven techniques, mainly the integration of development tools via a Meta Object Facility (MOF) repository backbone, the provision of automation by facilitating model transformations and code generation and by increasing the consistency and traceability of all development artifacts. An evaluation within NTT Data showed a development effort and time reduction of more than 50%. The main lessons learned form this case include the necessity of model-to-model transformations, the need to preserve advantage of existing development process and tools landscape, the requirements to care about configuration and change management from the beginning and requirements to support also non modeling tools.

1. Introduction

Software and systems development techniques and tools have dramatically increased their capabilities during the last years. New products and paradigms are coming down the road, claiming to have the potential to increase particular activities or steps in the systems modeling and development process of a software development company. However, those new methods or tools often require the adaptation of the modeling and development process established in such a company. Changes or adaptations in the development process definition can easily be done in small, agile companies who might always adapt their systems development to each customer project. In contrast to them, for a large system integration company like NTT Data the situation is different, because they have their own software development processes and design techniques. Such processes and techniques usually construct their competitive advantage and are widely accepted by their employees. Therefore, in such circumstances, new methods and tools have to adapt to the modeling and development process as well as to the existing development tools landscape. That justifies why large organizations usually cannot just apply out-of-the-box solutions and do always require an integrated and customized, process specific version of new system development techniques and tools. Accordingly, solutions need to be provided to such large organizations that allow both keeping the existing development process and tools on the one hand and introduce new development methods and techniques on the other.

The provision of such solutions by IKV++ Technologies AG is based its core technologies called medini. medini comprises of a set of tools and components around the Model Driven Architecture (MDA) [6] initiative. In MDA, well-defined models replace programming code as primary development artifacts. Based on a clear semantic foundation using Meta Object Facility (MOF) [5] system modeling and development methods and tools are seamlessly...
integrated to form open, extensible environments - customized and specialized to the targeted modeling and development process. An overview of a typical solution based on medini technology is shown in Fig. 1. Multiple distinguished methods and tools are used for the different activities and steps in the development process. Enabled by medini, these methods and tools can run in a seamless environment. This environment takes care of the storage and management of all development artifacts in a repository infrastructure which forms the backbone for integration. The demanded automation and consistency can be delivered by model transformation and consistency checking technologies as they are also part of medini. For model transformation, medini in its current production version provides a MOF transformer skeleton generator and for consistency checking it contains a full Object Constraint Language (OCL) 2 [3] compliant constraint engine which runs on MOF compliant models.

Fig. 1 Development tool integration based on MOF repository technology

The medini technology has been applied for the automation of one of the system development processes applied in the Public Business Unit of NTT Data. The development process aims on the provision of Web-based information systems which are running on a NTT Data specific platform which is an abstraction layer for J2EE/Java [9] technology. The result of this work is the AMEDATO solution - it integrates modeling tools and techniques applied by NTT Data and automate designated development phases and steps.

In the section 2 of this paper we talk about the starting conditions for the process optimization task that lead to the AMEDATO solution. The first version of the AMEDTAO solution which has been provided mid 2006 and its technology foundation is described in section 3. This first release of AMEDATO has been evaluated by NTT Data in a project context and the results of this evaluation are summarized in section 4. Some of the recent extensions to AMEDATO are explained in section 5. The lessons which we have learned during the execution of the project are provided in section 6. They will guide the further work on technology improvement as well as further solution projects for other companies.

2. Starting Conditions

The AMEDATO initiative stared in December 2004. At NTT Data, the corporate wide development process for business modeling, requirements definition, design and coding of large information systems was precisely defined in a bunch of documents. The process definition was widely used in most of NTT Data's development projects. The process has been constructed based on a lot of experiences from large scale software developments NTT Data has done so far now. This process definition documents include the detailed description of the to-be produced development artifacts. Furthermore they describe how the development information is exchanged between the development phases. Mostly, this exchange was realized in practice by text documents and spreadsheets, but also UML diagrams were already in use for some artifacts like Use Cases. The development process and main phases are sketched in Fig. 2.

Fig. 2 Document centric development process as target for automation

Although the process description document contained already a specification of transformations rules (for example the transformation of design classes on platform specific Java code was explained) the automation degree was very low. In fact, some small
tools existed which could extract information out of spreadsheets and transformed them into code. The usability of these tools was limited in the sense that the information was not combined, changed or enriched as for usual transformations but just extracted and for example inserted into XML documents or one-to-one spit out to Java code. Besides the lack of automation, a lot of development resources have been wasted for keeping all development artifacts up to date. In other words, the consistency between the development artifacts at the various steps in the development process has been identified as the main problem which required a solution. Change requests, e.g. if a customer contacts NTT Data to change the development goal model or requirements specification, do usually result in inconsistencies between the development information contained in different documents. Since there are no automatically managed references - which we call traces - between the various information elements it has been hard to maintain consistency by manual maintenance work. Moreover, changes did have to be manually propagated through all the subsequent phases in the development process without automation support. This approach has been especially error-prone in the case that changes do effect early phases like requirements or business models.

NTT Data therefore requested a significant improvement of the productivity by automation and also support for consistency checking and maintenance. Furthermore, they wanted to preserve the process itself and also the tools used so far should be kept in place. If a change to the tools landscape would turn out to be necessary, it was required that the resulting solution should not rely on expensive new tools.

The overall business modeling, requirement capturing, system planning and realization process is made up of 15 different phases, targeting all of them in one shot turned out to be not efficient. Therefore it was decided to firstly target the phases Use Case Design, Screen Flow Design, Control Design, Detailed Class Design and Coding. This selection was natural due to the fact that they showed a high potential for automation. After successfully finishing this first step, the solution should be extended to the more upstream phases and the testing as well.

In technology terms, at the starting point of the AMEDATO project, the medini base technology at IKV contained a MOF repository generator which was able to completely generate model repositories from MOF 1.4 metamodels in C++ language including remote access via CORBA [4] and XMI [8] technology. Furthermore, an OCL constraint checker was available, which can generate C++ code for the constraint evaluation with high performance as well as to interpret them at runtime.

3. The AMEDATO Solution

For the provision of the AMEDATO solution, the “medini process pattern” of IKV++ Technologies AG has been applied. This methodology has been developed to interweave the concept of Model Driven Architecture into existing or to-be defined workflows (e.g. development processes) of customers enabling them to benefit from the MDA advantages without having to recreate their existing development cycles from scratch. Making the entire MDA aspects - specifically the application of abstract techniques such as MOF, OCL etc. - completely transparent to the customer during the joint development and collaboration is one of the key features of the medini way of helping applicants to become more efficient without having to worry about the intrinsic details of the underlying technologies applied. According to the medini process pattern, the optimization and automation of development processes begins with an in-depth analysis of the currently applied process, the applied development tools, the artifacts to be produced in each development phase and their various relations. This analysis work leads to a metamodel or metamodells for development artefacts. These metamodels are used later in the process as the heart of the tools integration and automation environment: they are the main source for providing model repositories which do form the backbone of any tools integration and automation environment. According to the medini process pattern, under careful consideration of the as-is methods, techniques and tools situation, the selected development tools are subsequently customized and tailored to the process. They are connected to a model repository infrastructure obtained by application of the MOF standard to the metamodel. Model transformations, consistency checking and maintenance as well as traces management between development artifacts are realized by leveraging the detailed information in the model repositories. The following subsections do give an overview on how the medini process pattern has been applied to AMEDATO.

3.1. Process Analysis and Metamodel Definition

The development process applied in NTT Data is defined in terms of the supported development phases and the artifacts to be produced in each of these phases. It is described in a set of documents available to the
developers. For each phase, a set of tools or notations is also defined which is applied to let the developers produce the necessary information. During the process analysis done according to the Medini process pattern, it is the task to understand the development artifacts in detail. This deep understanding by the development process analysts is necessary to formalize the gained information in a set of metamodels since each and every aspect and property of a development artifact element has to be precisely defined there to let the resulting set of metamodel definitions be a mirror of the actual development process.

As the first target for automation of the NTT Data development process, 5 phases have been selected ranging from the system use case description to the application coding. The AMEDATO metamodel defined during the analysis phase presently contains the detailed specification of the development information for these phases and their relations. Those relations play an important role for the subsequent definition of traces, consistency rules and model transformations as well as for code generation. The metamodel is based on a restricted subset of the Unified Modeling Language 2 (UML) - the "Generic" package of the AMEDATO metamodel is actually based on the "Kernel" package of the UML 2 metamodel. It contains both structural and behavioral definitions like classes, use cases, activities and basic actions. The decision of using the UML 2 metamodel as foundation for the AMEDATO metamodel has not been taken for tooling reasons, but because the elementary concepts of the NTT Data development process do have a number of similarities with the UML 2 core concepts.

All elements in the AMEDATO metamodel which are specific for the NTT Data development process phases are inheriting from elements in the package "Generic". To define these specific elements, the documents and artifacts produced by NTT Data in real development projects together with the process description document have been analyzed (Fig. 3). In the following we give a few examples of how the detailed metamodel has been defined based on the analysis of existing artifacts from real developments:

- the system planning and development is based on use case driven requirements capturing, leading to the inclusion of the use case concept in the AMEDATO metamodel,
- the detailed properties of use cases going beyond those defined in the UML 2 metamodel, which are used for system planning and for development, are usually specified in large excel tables. The details of the columns of these Excel tables are candidates for properties of a metaclass for the development process specific use cases,
- the development process includes an activity to describe the structure of user interface pages (so called screens) and the flow of user interface pages depending on user input or system output (so called screen flows); these concepts have been included in the AMEDATO metamodel as specialization of the UML 2 activities package,
- the detailed concepts used in the implementation model for a system under development have been included in the AMEDATO metamodel as specializations of the UML 2 concepts class, interface, method, behavior etc.

![Fig. 3 Example of a development artifact specified in Microsoft Excel](image)

### 3.2. Notation Definition and Tool Customization

Now that the AMEDATO metamodel precisely defines all modeling concepts and their relations in the NTT Data development process, it is the basis for defining modeling guidelines and rules, model transformations between the various parts of an AMEDATO model belonging to the different development process activities covered and it also serves as the specification for the AMEDATO model repositories to manage the artifacts of the development process. Besides, the metamodel declares an abstract language for AMEDATO models due to the fact that all the concepts are declared and the associations in the metamodel precisely define how the concepts are related to each other. It should be noted that the metamodel does not define a concrete language, i.e. how concrete models are to be visualized to the user (the NTT Data engineers and developers). This visualization can be done in several different manners. For example, there can be a
textual, tabular based presentation for the AMEDATO use case concept. The same use case can also be presented in a UML 2 notation. This flexibility provides the freedom to declare a notation which fits the needs of the target user best. Moreover, multiple notations for the same metamodel element can be defined thus allowing multiple tools to be used for editing them. As said, the NTT Data development process contained the definition of notations to be used in each process phase already. For several artifacts, graphical notations have been used. Based on the MOF metamodel, the AMEDATO architects defined notation elements for each metamodel concept. These newly defined notation elements have been drafted to be as near as possible to the already used graphical notations. As implementation basis, UML 2 profiles have been created for these graphical notation elements as well as tabular notations to serve the non-UML users.

As an example, NTT Data has been using a graphical notation for the so called screen flow design as introduced above (Fig. 4). As said, this development artifact defines the screens of an user interface, the controls available on each screen and the transitions between screens based on the controls and the system output. In the AMEDATO metamodel, the concepts of screen, screen transition etc. have been introduced as specializations of modeling concepts from the UML 2 Activities package. The concept of a screen became a specialization of the concept Action known from the UML 2 language; the concept screen transition became a specialization of control flow between actions and so forth. The newly introduced graphical notation for the mentioned concepts uses actions and control flows between actions as well. This new notation looks quite familiar to the NTT Data engineer, but each element has a quite precisely defined semantics.

After having finalized the analysis phase, metamodel definition and notation definition phases, appropriate front-end tools have been selected and customized to be used by NTT Data engineers. For the graphical notations, the UML 2 tool Enterprise Architect of Sparx Systems has been adopted. Apart from the integration of the tool into the AMEDATO infrastructure, which is described in the subsequent sections, the tool appearance has been completely customized to support only those notation elements that are defined specifically for AMEDATO, using the built-in profile mechanism.

For the tabular notations which will still be used in the AMEDATO solution, a hybrid approach has been taken. For some of the tabular development artifacts, specific dialogs and tabular input fields have been added to the customized UML 2 tool. As an alternative tool for tabular notations, which still does play an important role with the NTT Data development process, the Microsoft Excel application serves as front-end, for editing generated programming language code, the eclipse environment has been integrated.

3.3. Transformation Definition

The analysis of the relations between development process artifacts lead to the detailed definition of relations between metamodel elements representing those artifacts. These metamodel relations do play another important role: many of such relations could be automated, i.e. artifacts of one development process phase can be created partially or completely from elements of another process phase. To achieve this, the candidate relations have been identified and so called model-to-model transformers have been realized. In AMEDATO, the approach of defining multiple sets of model transformation rules was preferred instead of having one large set of model transformations that works like a magic machine. Small transformation steps, understandable for the user, do support the acceptance of this technology by the engineers and developers. The whole solution therefore works in a way that the user
provides a number of definitions in the model in one step of the development process; afterwards a set of small transformations from one step in the process to the next is applied, and the user provides certain additional properties and aspects of the model elements created by these model transformation steps. These enriched elements are taken by the next set of model transformation rules to produce the elements of the subsequent step in the process and so on. The main advantage is as said the understandability of the transformations by the user. These transformers do not only create elements of a model, but more importantly merge the result of a transformation with the existing parts of the model. That means that the re-invocation of a transformation step due to changes in the transformation source does not just overwrites the target elements, but merges the result of the re-invocation together with the existing elements in the target parts which possibly have already been enriched by the user and thus cannot be just discarded. The concepts for automated model merge do show another interesting aspect: the same approach has been used in AMEDATO for managing parallel editing work of multiple engineers regarding the same model. Besides, each execution of model transformations stores traces information. Traces play an important role for aspects like change management, consistency checking of models and system code as well as impact analysis.

After the first phase of AMEDATO, the following model-to-model transformations exist:

- the skeletons for conceptual screen flow designs\(^1\) are created from use case definitions,
- automated transformers produce detailed screen and screen flow definitions from conceptual screen and screen flow designs,
- the detailed implementation design is produced from detailed screen flow definitions and database definitions imported during transformations execution\(^2\),
- the configuration of controls and views is produced as a kind of Model-View-Control (MVC) diagrams from detailed screen flow designs.

Besides the mentioned model-to-model transformations, a very large part of the source code for the target system is produced by code generators. The code generation rules are defined based on the metamodel part that described the detailed implementation design. Thus, the code generation rules produce a one-to-one representation of the detailed implementation design in the source code. The problem with re-invoking code generators onto previously produced code fragments appears similar to re-invoking model transformers. An intelligent code merge facility has been integrated with the code generators to avoid the unwanted effect of overwriting and thus discarding existing code.

Fig. 5 Tool architecture of AMEDATO

3.4. Tool Architecture

The tool architecture of the AMEDATO solution (as of April 2006) is shown in Fig. 5. Out of the AMEDATO metamodel, the AMEDATO repository has been created automatically using the medini technology. This technology creates MOF compliant repositories supporting model versioning, configuration management aspects as well as remote interfaces. Furthermore, a transaction mechanism takes care of concurrent access to AMEDATO models. A persistency layer does support the storage of AMEDATO models in databases. The modeling tools as well as other front-end tools have been connected to the repository facilitating the mentioned remote access. The customized Enterprise Architect tool connects to the repository and is used for all graphical modeling tasks, the ER-Studio tool is used for database design; those designs are imported into the AMEDATO repository during the model-to-model transformation from the detailed screen
and screen flow design to the detailed implementation design. The eclipse environment has been extended with a specific plug-in to allow for code generation from AMEDATO detailed class designs, the code generation itself is realized by facilitating the Java Emitter Templates (JET) technology connecting to the AMEDATO repository as well. The whole documentation can be generated from the artifacts stored in the repository by again applying JET with the DocBook technology as target.

3.5. Additional Functionality

A number of additional functionalities have been provided to maximize the value of the tools for the development in NTT Data's projects. Some selected features of those functionalities include:

- The commonly known profile mechanism from UML 2 has been integrated and propagates profile definitions and profile applications to the AMEDATO metamodel and therefore to the AMEDATO repository. This feature allows for the customization of the AMEDATO metamodel and the instant support of the customized AMEDATO modeling concepts in the connected modeling tools. For the specific use of AMEDATO in a development project, a profile definition may provide additional properties for elements of the AMEDATO metamodel or may define additional relations not covered by the general AMEDATO metamodel. Based on these extensions, the profile mechanism automatically supports editing/visualization capabilities for the extensions in the front-end tools as well as enables the storage of values of such extensions in the AMEDATO repository. Profile definitions can be facilitated for additional model transformation rules, specific code generation rules or fine grained consistency checking rules.

- A model validation engine based on the Object Constraint Language (OCL) 2 has been integrated with the repository. The engine is facilitated for all model consistency checks. These consistency checks do rely on consistency rules, which are defined as OCL constraints and can be invoked on the whole or parts of the model. The engine interprets OCL, therefore it is possible to add new consistency rules on the fly. In terms of usability, the constraint definition is hidden from the engineers view, he only sees the meaning of the consistency rule and may invoke the rule checking based on such meaningful description. Additionally, also impact analysis rules are expressed as OCL constraints and can be executed by an user based on a meaningful description as well.

- A template based engine for the export of repository elements to Microsoft Excel spreadsheets and the import of elements from spreadsheets is integrated in the tool environment. A user can specify an Excel import/export template using a simple language based on the AMEDATO metamodel. Such templates can be applied to models stored in the repository, the results are used for documentation purposes and for communication with external parties. It should be noted, that the definition of a import/export template is based on the AMEDATO metamodel as well: the contents of a single cell, a row or a column in the spreadsheet to be created or read is defined by a kind of select statements upon metamodel elements - resulting in the production of spreadsheets containing the values store in the repository for the model the template is applied upon (export direction) or in creating/manipulating the specified elements in the repository for a spreadsheet the template is applied upon (import direction).

4. Measurable Effects

A common problem with the introduction of new development environment into a company is that at is necessary to clearly identify the benefits and the expected Return of Investment. Otherwise decision makers cannot be convinced of investing in new technologies. Therefore clear measured effects need to be proven. Even if such effects can be shown clearly, the introduction of model driven technologies into large organizations still comes with a lot of uncertainties and risks; a high level of risk acceptance of such organizations is required. Moreover, the introduction of such technologies cannot be done from one day to the next - it is a long, possibly expensive process and careful consideration of the acceptance by engineers, possible cost saving or development time reducing effects as well as necessary investments into the long-term roll-out seems to be required.

NTT Data has decided to prove the benefits of the AMEDATO solution by performing a virtual development project tool supported using AMEDATO. For that case, a typical mid size project was selected, which targeted the development of a web-based system for a division of the Japan Government. The basic design for that project has already been completed and

\[\text{1. There are not two, but just one such template for both export and import direction, but the evaluation semantics for each direction is different.}\]
all the necessary material from this phase was available. NTT Data engineers performed the modeling and development with AMEDATO starting with the system use cases, continuing via Conceptual and Detailed Screen Flow Design, Control Design and Detailed Class design until the Java code was generated. They completed the Java code by the necessary business logic manually afterwards. The degree of automatically generated code has been measured to be 76%. (This degree depends on the requirements of the project but the selected project was a typical one so a similar result can be expected when applying AMEDATO to other projects.) The engineers provided and performed the unit test cases for the Java code. In addition, they generated all the documentation with the AMEDATO tools automatically.

For comparison, NTT Data asked for a offer from one of its typical sub-contracted companies. Such a company offer with the same items (Java implementation, documentation, unit tests) and with the same starting point (basic design specification) showed more than double of the effort compared to the project team that applied AMEDATO. Furthermore, the NTT Data engineers finished the project in the half of the development time estimated by the sub-contractor. According to the measured results, it is possible for NTT Data to achieve a Return of Investment (break even) by performing approx. 7 typical size projects with the AMEDATO toolset in the style described before.

To estimate the real potential, we have to consider, that the above case only shows the benefit in case of pure forward engineering. In the more realistic situation where the requirements may be adapted or other change requests appear during a development project, the benefit of AMEDATO will be much higher since the consistency is automatically managed. Further benefits are expected if the test and business modeling phases are also integrated into AMEDATO as it is already under development. The necessary training effort for the engineers is acceptable for NTT Data, since the process itself hasn't changed too much and the notations and development tools applied in each phase have been re-used or considered when replacing them by new tools.

5. Recent AMEDATO Extensions

5.1. AMEDATO Support for Business Modeling

Before a development project based on AMEDATO is started, a detailed communication and negotiation takes place between NTT Data and their customers. This process is supported by a detailed business modeling method called MOYA developed by NTT Data. With this modeling process, the current business situation and the anticipated business situation after the introduction of some new system can be modeled and analyzed. The method is based on the Soft Systems Methodology by Peter Checkland [10] with some modifications/extensions (e.g. goal oriented analysis instead of Conceptual Model). Throughout the process, the different stakeholders of the business and the opinions/goals of them are identified, and the as-is business situation and its transition to the anticipated business situation is analyzed. Furthermore, the connection between the business modeling result and the system specification and development with AMEDATO is supported. AMEDATO has been extended to cover also the MOYA business modeling method. The basic goal of that extension is to provide a tool environment to optimally support the application of the business modeling method. The single-source principle based on the AMEDATO repository is applied such that the different artifacts resulting from a business modeling process application are managed in a repository, similar to the AMEDATO tools for the system development. Rich traceability, the consistency checking with constraints and the support for managing model configurations plays an important role in the extension.

Due to the intensive relations between MOYA and the other activities in the overall system development process it was decided not to invent a completely new architecture for AMEDATO-MOYA. Instead an integration of the MOYA tools into the overall AMEDATO-X tool architecture has been planned. AMEDATO now provides an integrated tool environment for the MOYA process and the development process. MOYA technically consists of several steps to capture the current business situation, to derive the anticipated business situation and to help in system planning. For all of these steps, graphical modeling techniques are applied. The notation for modeling is based on UML 2. UML 2 structured activities are used for the business process modeling, UML 2 use cases are used for system planning support and structure diagrams are used for goal modeling and business resource modeling. Profiled versions of other elements of the UML 2 language are used to express specific concepts of MOYA, e.g. the specific properties of MOYA model elements are captured as special properties of the UML 2 elements used to graphically represent MOYA concepts.

In more detail, the AMEDATO extension for MOYA consists of the following components:
The AMEDATO repository extension for MOYA -

The repository is used to store all elements of a MOYA application, e.g. all process model elements, resource and goal models and so on including diagram information and trace relations to other elements in a MOYA model are managed by the repository. Everything that is modeled or somehow produced with front-end tools is stored in the front-end tool independent AMEDATO repository. The repository is automatically generated out of the AMEDATO metamodel extension for MOYA using the medini technology. The metamodel extension for MOYA is again based on the UML 2 core packages and extended by definitions specifically for MOYA. In the MOYA business modeling process, the maintenance of such relations (e.g. between identified stakeholders, their opinions and the goals for the business improvement) is of high importance. The management of such relations can be executed inside the AMEDATO repository very performant.

The AMEDATO IDE extension for MOYA is used to manage the tool application, the repository as well as the connected front-end tools. In terms of the tool architecture, the IDE also contains a multiple version repository that is used by one single modeler.

The Enterprise Architect UML 2 tool customized for MOYA - the Sparx Systems Enterprise Architect tool is a generic, standard UML 2 tool and therefore does not know about the specific semantics of the concepts used in MOYA. The tool has to been completely customized for its application to support the MOYA process. This customization has been done by the provision of an UML profile and a Plug-In for the tool to provide MOYA specific modeling and editing support. The Plug-In synchronizes the visualization of MOYA models in Enterprise Architect with the storage of those elements in the AMEDATO repository. The Plug-In accesses the Enterprise Architect specific model representation and uses the remote API of the repository to load and store model elements. It should be noted here that an active synchronization style is realized meaning that if a model element is added/changed/deleted inside the Enterprise Architect model, it is at the same time added/changed/deleted in the repository too. Furthermore, the correctness of the static semantics of MOYA models stored in the repository can be checked instantly, especially during editing of the elements.

The Microsoft Excel Import/Export Facility - NTT Data uses spreadsheets to exchange information with their customers. These spreadsheets contain artifacts produced during MOYA applications to communicate with people in the customer's organization. To support that behavior in the AMEDATO extension for MOYA, an Excel Import/Export facility has been provided.

The MS Excel based Traceability Matrix - the AMEDATO extension for MOYA does also support the creation of a traceability matrix in a tabular notation, using MS Excel. This matrix allows to read all traces which are defined between elements in MOYA (or more general in AMEDATO models), to edit those traces and to import edited trace matrixes back to the repository.

A specialized version of the constraint checking engine as part of the medini metamodeler repository core libraries has been integrated into the AMEDATO extension. With this engine it is possible to evaluate arbitrary constraints defined in OCL against the elements in the AMEDATO repository. This engine is used for the evaluation of consistency rules for AMEDATO - and especially MOYA - models. The engine reports all model elements that violate the consistency rules.

5.2. Integration of Web-Design

The first version of the AMEDATO solution was able to generate the initial GUI of the application based on Java Server Pages and Struts out of the information in an AMEDATO model. This kind of code generation has been replaced subsequently due to the fact that the "art work" - the production of the HTML code - and the functional work - the definition of the screen structure and screen flows - are usually parallel tasks in a development project. In particular, the usual development process foresees that a professional user interface designer works in parallel on a nicely layout and styled user interface (art work). Back then, engineers manually merge the JSPs resulting from the detailed screen flow definition and the manually created html pages by the GUI designer. This process turned out to be difficult if the complete auto generated JSPs as a result of the application of the AMEDATO environment are assumed. As part of the AMEDATO environment, tool support has been realized to support for the synchronization between the art work and the detailed screen and screen flow designs as well as the merge of the art work and the generated JSP.

6. Conclusion

The whole AMEDATO solution is based on precisely defined MOF metamodels. Besides its technological role, the metamodeling approach turned out to be a beneficial discussion basis for further process
extensions and enhancements although it was originally just foreseen for an internal automation purpose. The often mentioned argument that metamodels are for academic purposes only turned out not to be true, after some learning process, it was easy and very productive to discuss issues in the AMEDATO team mainly based on the metamodel definitions.

Another point is that it is simply impossible to radically change development approaches and processes in large companies. Also, the processes shouldn't be radically changed because they are a kind of "know-how" the company has. A company usually doesn't want to waste their competitive advantage although the new techniques offer some of potential of new advantage. If one wants to introduce new software engineering techniques into existing development processes and teams, it is extremely important to keep the current advantages which the existing process has and to introduce the new techniques at carefully identified points of the process. For that identification, we used an approach of collecting painful situation descriptions - developers and modelers know in which situation they feel an extraordinary difficulty or spent a lot of time and efforts to achieve a certain result. The analysis of these painful situations was one of the reasons for the acceptance of the new tools and approaches. The mentioned keeping of the development process also holds for re-using already existing tools landscape.

Even if such a careful integration of model driven technologies into an existing process is chosen, a large company doing so still bears a lot of risks. While working on the AMEDATO solution, non-functional issues such as usability, understandability and maintainability did play an extraordinary important role. The technologies as such are not convincing, but their integration into a smooth, usable, cool environment like AMEDATO turned out to be one success factor.

Model-to-model transformations do play an important role in the AMEDATO solution. Many small transformations turned out to be more applicable than one big fat magic model to code transformation. In conjunction with model transformations, aspects as model merge, versioning and navigability (based on managed traces) are a key factor. As an extension to the medini technology, IKV realized a QVT engine that implements the Relations part of the standard. In the next step in AMEDATO, this technology will be integrated into the solution to improve the flexibility of the transformation rule definition.

The whole business modeling, requirements capturing and development process of NTT Data consists of 15 different steps. Accordingly, 15 distinct kinds of models can be identified. This makes the very simple categorization of models into CIM, PIM and PSM as promoted by the MDA difficult to be translated into the real world. We suggest to not specifically discuss in such model categories, but to simply talk about various models at different or same levels of abstraction that have some relation with each other.

7. References


