# Recommendations for Impact Analysis of Model Transformations
## *From the Requirements Model to the Platform-independent Model*

Dmitri Valeri Panfilenko[1], Andreas Emrich[1], Christian Meyer[2] and Peter Loos[1]
[1]*DFKI GmbH, Saarbrücken, Germany*
[2]*t.e.a.m. Unternehmensberatung AG, Eschborn, Germany*

Keywords: MDA, Impact Analysis, M2M Transformations, Recommendations.

Abstract: Model transformations are the core component of MDA. They make it possible to transform models between different levels of abstraction, which allows the implicit in-built knowledge to be passed on from domain experts to the IT professionals. What is not considered by the OMG, are the consequences that changes at each level cause to the other MDA levels, which could be estimated through impact analysis techniques. For example, if the course of a procurement process in a company is to be changed, this would be performed by the proper experts at the technical level. However, it is difficult at this time to estimate the resulting changes at the following adjacent levels. This shortcoming needs to be addressed and proper recommendation support for the impact analysis of model transformations has to be elaborated.

# 1 INTRODUCTION

This paper emphasizes the extensibility of MDA. Extensibility is present partly due to the fact that MDA relies on a variety of other OMG standards (OMG, 2003; OMG, 2011a; OMG, 2011b), which are usually also very extensive. Another reason for this lies in the vague definition of the OMG. As pointed out by critics, such as Greenfield et al., 2006 MDA focuses too much on the platform independence as its main aspect, and leaves many complex aspects of software development largely unanswered.

This paper seeks to combine techniques from impact analysis with recommender systems approaches by using the cost models of the impact analysis as design criteria for recommender mechanisms. The paper will deliver a blueprint for how to combine these techniques and to enable a method of modelling support for MDA tooling.

The paper will shortly outline the shortcomings of contemporary approaches and tools in this area in section 2 and will then explain how impact analysis can be used for model transformations in section 3. Section 4 will discuss appropriate cost models for transformations and shows how to integrate these with a recommender systems approach for model transformations. The paper will conclude with a summary and an outlook on future implementations and the potential benefits of the outlined approach.

# 2 RELATED WORK

## 2.1 Approaches

There is a number of approaches for impact analysis of model transformations and accordingly different surveys for their classifications (to name a few: Czarnecki and Helsen 2006; Mens and Van Gorp, 2006; Sendall and Kozaczynski, 2003). The most interesting feature-based survey has been made by Czarnecki and Helsen, in which they have drawn the following types of the model-to-model transformation approaches: direct manipulation, structure-driven, operational, template-based, relational, graph-transformation-based, and hybrid.

All of the presented approach types have their pros and cons: direct manipulation is very low-level and requires user interaction; structure-driven is well applied to generating Enterprise Java Beans along with database schemas from UML models, but unclear whether they are applicable to other kinds of applications; template-based are particularly well-suited for code generation and model compilation, although not providing traceability out of the box; relational seem to be most applicable to model synchronisation scenarios, though may experience


Valeri Panfilenko D., Emrich A., Meyer C. and Loos P..
Recommendations for Impact Analysis of Model Transformations - From the Requirements Model to the Platform-independent Model.
DOI: 10.5220/0004971604280434
In *Proceedings of the 16th International Conference on Enterprise Information Systems* (ICEIS-2014), pages 428-434
ISBN: 978-989-758-028-4

performance draw-backs depending on the constraints to be solved; graph-transformation-based are without doubt the most theoretically sound though lacking the coverage of all of the possible model transformation landscape; hybrid solutions are mixing the other approach types depending on the given transformation scenarios.

The existing model transformation approaches describe extensively the features of the supposed changes to be made. At the same time, there is a seeming lack of process description, a kind of workflow behind the scene that would provide a guideline for the stakeholders for assessing the changes made to the model and their propagation to the adjacent modelling levels, which draws the attention of this paper.

## 2.2 Tools

There are four tools we provide overviews for in this paper, namely *AndroMDA*, *PowerDesigner*, *Rational* family, and *Modelio*. For each of these tools, the feature highlights are addressed first and then the support for modelling on different MDA-levels as well as for M2M-transformation and its impact analysis are questioned.

*AndroMDA* (pronounced "Andromeda") is an open source tool supporting many features including UML modelling and deployment onto different platforms. AndroMDA is basically a transformation engine offering modelling support for PIM- and PSM levels as well as transformation to code. The impact analysis features are not explicitly supported or mentioned.

Sybase provides a commercial modelling tool *PowerDesigner* for enterprise architecture modelling, which supports several modelling techniques on different levels of abstraction as conceptual, logical and physical. Overall, it is a powerful tool offering modelling on the CIM-, PIM- and PSM-levels as well as a bridge to the execution environments through support of the BPEL export, which in addition supports impact and lineage analysis of the certain model artefacts.

IBM's *Rational* family is a well-known commercial tool family supporting modelling of the different aspects of the enterprise architecture with established standards like UML targeting different programming languages. This tool offers support for modelling on CIM-, PIM- and PSM levels with code generation to different programming languages and some support for horizontal and vertical traceability, as well as a defined impact analysis workflow.

*Modelio* is a famous commercial modelling tool with explicit model-driven development support, which offers support for modelling on CIM-, PIM and PSM-levels, for code generation to different programming languages and some support for role management in the team solution as well as dependency diagrams for impact analysis of the models.

# 3 IMPACT ANALYSIS OF MODEL TRANSFORMATIONS

This section elicitates the obstacles to impact analysis of model transformations. In particular, importance has been put on the fact that the whole information contained in the source model should be preserved during a transformation to a target model and all the transformations should also be isomorphisms or at least bijective homomorphisms.

## 3.1 Challenges of the Model Transformations

The following challenges during conducting of the model transformations should be considered and be taken into account. The typical solutions are stated in each case (Stahl et al., 2007; Kleppe et al., 2003):

- **Deal with Quantities** – If there is a need for a function of all or only certain elements of a list that satisfy a predetermined condition, this has to be realized somehow. In Java this can be done using a loop that goes through the elements one by one, which might be too cumbersome when the number of elements is large. Typical transformation languages allow this to be implemented significantly easier through involvement of the declarative programming statements (Becker, 2009).
- **Deal with Cycles** – There are already difficulties if two artefacts of the source model, A and B, refer to the same object C in the target model, which could be solved by storing the target model artefact $C^t$ in a cache, checking that it already exists and therefore not regenerating it. It gets even more difficult when there are cycles. This means that model X references model Y, which itself refers back to X. This cannot be solved as in the first case with the cache, since none of the states has been initialized and thus none of them stored in the cache. Transformation tools solve this problem using model traces (Balzert, 1999).
- **Debug Capability of the Transformation Language** – It is quite possible that a small change in the source model causes enormous changes in the

target model after the transformation without any obvious reason (Bohlen, 2003; Brown, 2004). Thus, the exact process of transformation needs to be reconstructed in order to find the exact cause, for which the modelling tool requires certain debugging functionality.

- **Identify Incremental Transformations** – In the case of adding information specific to the target model, this information should be preserved after the regeneration or new transformation. In Stahl et al., 2007 they consider this to be of minor importance because the technical effort should be enormous and practice would show that this requirement plays no decisive role.
- **Support Bidirectional Transformations** – Those transformations impose special requirements: they can be executed not only in one direction from the source to the target, but also vice versa. There are basically two ways to define a bidirectional transformation. Firstly, there may be a collection of transformation rules which are applicable in both directions. Secondly, there can be two separate collections of rules, each of them representing the inverse of the other (Beltran, 2007).

## 3.2 Model Transformation Types

A significant influence on the extent of anticipated changes and thus the expected costs have the connections between objects in the source and the target model. Thus, four different types of relationships can be distinguished (Berg, 2006; Jouault, 2006):

- **Injection:** a simple dependency of an element of the target model from one element of the source model.
- **Scattering:** the relations reach out from one source element to at least two elements of the target model.
- **Tangling:** one target element corresponds to at least two elements from the source model.
- **Crosscutting:** a mixture of scattering and tangling. At the same time, the target element is influenced by at least two elements from the source model and one of these source elements also influences another target element.

This nomenclature takes into account possible changes of the model artefacts from the source model to the target model. Other models transformation classifications are conceivable (Czarnecki, 2006). The focus of this article is set on exogenous bidirectional model transformations in context of MDA, specifically between CIM, PIM and further PSM levels.

## 3.3 Impact Analysis Process of Model Transformations

An overview of the process of impact analysis can be seen in Figure 1. In the first step, changes in the real world are observed. For example, this can be a process within the company to process customer orders, whereas the aim is to accelerate the processing of orders from existing customers. These changes have to be reflected in the highest modelling level of the company. Following the example above, this could be an EPC (event-driven process chain) for the process representation on the CIM level. The planned changes are sketched and conducted on this level, after which the actual impact analysis can begin using various techniques we abstract of at the moment (Arnold, 1993; Pohl, 2008). The results of the impact analysis reflect the changes in the currently existing system on the PIM level that have to be done and which consequences this would have. With aid of these results, the changes can be estimated, planned and conducted. The impact analysis might not end at the PIM level and could be propagated to the PSM level. In this case one has to be sure of the correct analysis in the first place, as the incorrect estimates would be taken into account for the further steps, resulting in the skewed model changes estimations. Apart from the sketched analysis estimations of impact, volume, explanations and execution more features could be integral parts of the comprehensive analysis: illustrations to the propagating changes in the system, access to the changes history, suggestions to the changes strategies, test of the system with the executed changes and so on (Arnold, 1993).

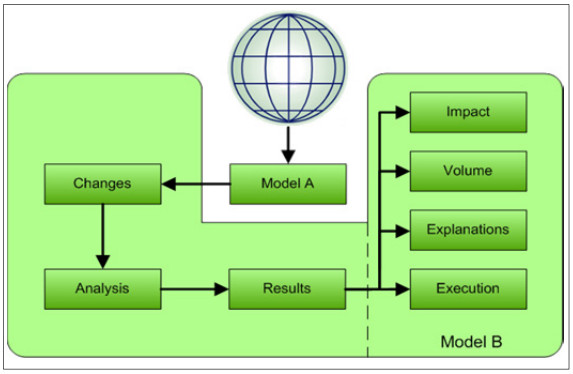


Figure 1: Proposed Impact Analysis Process of Model Transformations.

The ideal of the impact analysis for model transformations would be not to execute the transformation every time a change is made for the

obvious performances reasons depending on the size of the model, but rather to run a procedure in order to assess the largest possible impact on the models of the same, the adjusting and the consequent abstraction levels. For that we need means of assessing the impact due to changes made in the source model, which would give tangible cost estimates and thus get a solid basis for the decision making stakeholders.

As can be seen in Figure 1, changes have to be assessed according to their impact, possible interrelations among changes or the artefacts in question and how possible change action plans could be executed. In artefact-centric software engineering, a common technique in impact analysis to analyse, which artefacts are affected by certain changes in the software. According to Arnold and Bohner (Arnold, 1993) it is about "identifying the potential consequences of a change, or estimating what needs to be modified to accomplish a change". In literature three main types of impact analysis techniques are considered, namely: traceability, dependency and experiential analysis (Arnold, 1993). Traceability describes structural, global links among artefacts (e.g. "class Person implements Requirement R01), whereas dependency analysis deeply analyses the code and assesses whether a link constraint has been touched by a certain change or not. Even then, situations can occur, in which this cannot be determined automatically, so a fallback to experience knowledge involving real people – designers, programmers, etc. – is necessary to correctly assess these changes.

In order to assess different, possibly conflicting alternatives of a change, certain cost models need to be defined. Costs can be seen as an abstract terminology, as it could represent monetary costs or metrics such as TTF (time-to-fix).

The following figure depicts how this could be applied to model-driven architectures:

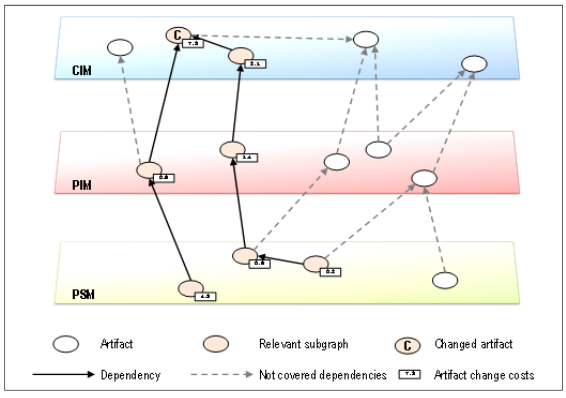


Figure 2: Impact and Cost Analysis for MDA Artefacts.

In this sketch, we have different artefacts distributed over the different levels of MDA: The CIM, PIM and PSM level. In this example, we mainly consider the traceability links as means for determining the artefacts affected by a single change. This network of dependencies is represented as a directed graph, e.g., as a code class may depend on a design document, but not necessarily vice-versa.

An artefact (C) is changed on CIM level. All dependent artefacts (depicted in orange) that have ingoing links to our artefact C or other dependent artefacts constitute the relevant sub-graph for impact analysis. In addition to Figure 2, artefact-specific analysis can reveal, whether certain fit criteria are met (e.g. if the proposed change is relevant at all for the dependent artefact).

Using this sub-graph, all relevant nodes (i.e. artefacts) and edges (e.g., transformation tasks, tests) have to be analysed regarding their costs. For simplification purposes, in our example we only show the costs related to artefacts.

A naive approach could add up the costs recursively, in order to determine the total costs of a change. However, this would not consider resource constraints (e.g. artefact A2 can only be changed by the developer that is on holidays) or opportunity costs (e.g., if we have a combined assessment of costs and time). In such cases, it is advisable to take this to a new level and to apply techniques from recommender systems research to this problem field.

The next section will explain how this basic impact and cost analysis can be used to provide recommendation support for model transformations in MDA.

# 4 MODEL TRANSFORMATION RECOMMENDATIONS

## 4.1 Cost Models of Model Transformations

Depending on the underlying cost model there should be different model changes graphs resulting from the defined model transformations and the previous model change traces.

Recommender system among others have two major phases in their recommendation process: filtering and ranking. Filtering refers to finding the appropriate candidate objects for recommendations, i.e., in our case model transformations. Ranking comprises the evaluation of the different candidate

objects according to multiple criteria and combines them in a so-called ranking function that computes a numerical score that denotes the relative usefulness of the given candidate object in the given recommendation scenario.

In our case, filtering could be realized by applying state-of-the-art model traceability and impact analysis approaches. They can determine which artefacts are impacted by a change and can derive appropriate change plans.

The ranking process would be rather more difficult. Multiple criteria can be applied for deciding which transformation alternatives are the best. General change metrics could be included, but also cost models for single artefact changes. Moreover, testing and integration efforts for artefact couplings could be a second degree impact that needs to be assessed in terms of effort and costs. This example could even be more complex, if an outsourcing situation is given, where a company has make-or-buy decisions, i.e., performing the change themselves or delegating this task to a third company. As these examples point out, the ranking of such model transformation alternatives is highly complex. At this point it is not clear, whether a generic ranking function could be developed as a best practice for the entire software industry, or whether we need to develop tailorable and customizable ranking models for such alternatives

## 4.2 Recommendations for Impact Analysis of Model Transformations

This paper's position refers to Gruhn et al., 2006, who describe six following use cases. These use cases would build a basis for the intended application of the cost models recommendations to the impact analysis of model transformations:

1. **Refinement** – An example is the case of transformation between PIM and PSM (Gruhn, 2006). Here, the PIM is extended to include platform-specific information and thus refines the PSM (OMG, 2003). Furthermore is a transformation from the analysis model to the design model is thinkable, which both reside at the PIM level.
2. **Abstraction** – An important use case for transformations (Gruhn, 2006). The development process is not a waterfall one but rather an iterative process which allows the review of artefacts at different abstraction levels. Thus, a backward step during the development might be necessary (Frankel, 2003). The transformation ensures that the abstract and the detailed models are synchronised with each other (Mellor et al., 2004).
3. **Migration** – The transformations are used to migrate a complete software system. It is necessary if the technical platform changes, in which case the transformation needs to produce a new system that can be fully utilised without any restrictions and will function properly. The more diverse the system landscape, the more difficult this change and therefore the transformation are (Gruhn, 2006).
4. **Refactoring** – This refers to „a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behaviour.“ (Fowler, 2008) Here, transformations are used to extract common features through generalisation or restructuring. The visible behaviour of the software, however, always remains unchanged.
5. **Optimisation** – Here, like in the refactoring, the functions of source and target are equivalent. Optimisation aims for the improvement of the space or run-time efficiency of a program. Unlike refactoring, though, optimisation is automated in its execution and based on a target platform (Gruhn, 2006).
6. **Changing the Presentation Form** – This covers the previously described M2T trans–formations, in which a graphical model is converted to a textual one (Gruhn, 2006). This type of transformation is most efficient when it takes place automatically, like for instance when changing class names in a UML diagram that immediately reflects in the underlying repository model (Mellor et al., 2004).

The directions of model transformation are summarised in Figure 3.

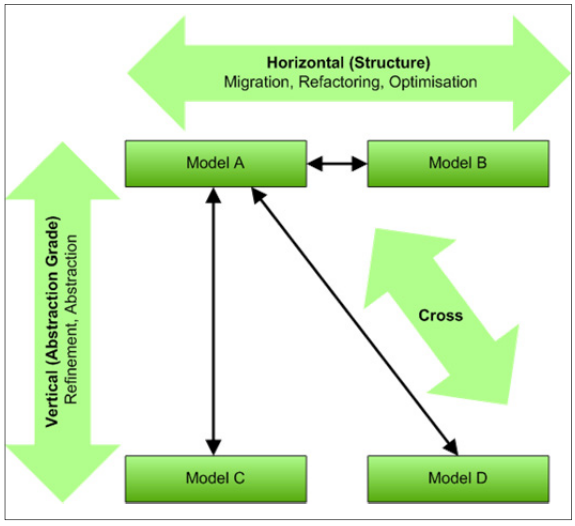


Figure 3: Transformation directions (according to Gruhn et al., 2006).

# 5 CONCLUSIONS AND FUTURE WORK

The first contribution of this article is the construction of a conceptual framework based on established research, which allows the expected changes to be categorised. Furthermore, it analyses what changes are to be expected as a result of transformations within and between model levels from the requirements model to the platform-independent models. This allows for better evaluation and planning of the expected consequences when future changes take place.

As a future work, it is promising to investigate impact analysis for transformations on the PIM level, such as by the transformation in PIM4Agents, which was proposed in the SHAPE project. Likewise, the impact of transformations on the PSM and the code level are also worth consideration.

An automatic process of impact analysis should be considered as the potential final goal of the research. This would enable it to deliver very early in an information change process, in which all business sectors are involved. In combination with the actually defined importance of observed effects, this would constitute a complete recommender system for deciding whether a change is reasonable or not feasible at all. Such a system, however, would require a lot of effort to be implemented in such a way that the impact analysis would produce accurate and reliable results in the real world. Before this point can be reached, however, there is still an enormous amount of research and development effort that needs to be done. This especially relates to the research that has to be carried out regarding ranking functions for model transformation alternatives and the alignment of multiple ranking criteria.

# ACKNOWLEDGEMENTS

This research was co-funded by the European Union in the frame of the SHAPE FP7 project (ICT- 2007-216408). The authors would like to express their acknowledgments to SHAPE colleagues.

# REFERENCES

Arnold, S. R.; Bohner, A. S., 1993. Impact Analysis – Towards A Framework for Comparison. In: *Card, N. D.: Proceedings of the Conference on Software Maintenanc*. IEEE Computer Society, Washington DC.

Balzert, H., 1999. Lehrbuch der Objektmodellierung – Analyse und Entwurf. Spektrum Akademischer Verlag, Heidelberg et al.

Becker, J.; Mathas, C.; Wilkmann, A., 2009. Geschäftsprozessmanagement. Springer, Berlin et al.

Beltran, J. C. F., 2007. Modellgetriebene Softwareentwicklung: MDA und MDSD in der Praxis. *J. Trompeter (Ed.)*. Entwickler. Press.

van den Berg, K., Tekinerdogan, B., & Nguyen, H. (2006, July). Analysis of crosscutting in model transformations. In *ECMDA-TW Traceability Workshop Proceedings* (No. A219, pp. 51-64).

Bohlen, M., & Starke, G., 2003. MDA entzaubert. *OBJEKTspektrum*, *3*, 52-56.

Brown, A. W., 2004. Model driven architecture: Principles and practice. *Software and Systems Modelling*, *3*(4), 314-327.

Czarnecki, K., & Helsen, S. (2006). Feature-based survey of model transformation approaches. *IBM Systems Journal*, 45(3), 621-645.

Frankel, D. S., 2003: Model Driven Architecture – Applying MDA to Enterprise Computing. Wiley, Indianapolis.

Fowler, M., 2008: Refactoring – Improving the Design of Existing Code. Addison-Wesley, Boston et al., 22nd edition.

Greenfield, J.; Short, K., 2006: Software Factories – Moderne Software-Architekturen mit SOA, MDA, Patterns und agilen Methoden. moderne industrie Buch, Bonn.

Gruhn, V.; Pieper, D.; Rötters, C., 2006: MDA® – Effektives Software-Engineering mit UML 2® und EclipseTM. Springer, Berlin et al.

Jouault, F., & Kurtev, I., 2006. Transforming models with ATL. In *Satellite Events at the MoDELS 2005 Conference* (pp. 128-138). Springer Berlin Heidelberg.

Kleppe, A.; Warmer, J.; Bast, W., 2003: *MDA Explained – The Model Driven Architecture: Practice and Promise*. Addison-Wesley, Boston.

Mellor, S.; Scott, K.; Uhl, A.; Weise, D., 2004: *MDA Distilled – Principles of Model-Driven Architecture*. Addison-Wesley, Boston et al.

Mens, T,; Van Gorp, Pieter, 2006: A Taxonomy of Model Transformation. *Electr. Notes Theor. Comput. Sci*. 152: 125-142.

Object Management Group (Ed.), 2003. MDA-Guide Version 1.0.1. http://www.omg.org/cgi-bin/doc?omg/03-06-01, Retrieved on 2013-11-04.

Object Management Group (Ed.), 2011a. OMG Unified Modelling Language® (OMG UML), Infrastructure Version 2.4.1. http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF/, Retrieved on 2013-11-04.

Object Management Group (Ed.), 2011b. Query/View/Transformation Specification – 1.1. http://www.omg.org/spec/QVT/1.1/PDF/, Retrieved on 2013-11-04.

Object Management Group (Ed.), 2013. Committed Companies and their Products. http://www.omg.org/mda/committed-products.htm, Retrieved on 2013-11-04.

Pohl, K., 2008: *Requirements Engineering – Grundlagen,*

*Prinzipien, Techniken*. dpunkt.verlag, Heidelberg, 2nd edition.

Sendall, S., & Kozaczynski, W., 2003. Model transformation: The heart and soul of model-driven software development. *Software, IEEE*, *20*(5), 42-45.

Stahl, T.; Völter, M.; Efftinge, S.; Haase, A., 2007: Modellgetriebene Softwareentwicklung – Techniken, Engineering, Management. dpunkt.verlag, Heidelberg, 2nd edition.