

Tool-Supported Systematic Model Assessment

Joanna Chimiak–Opoka, Gunnar Giesinger
Frank Innerhofer–Oberperfler, Bernd Tilg

Institute of Computer Science, University of Innsbruck
Technikerstrasse 21a, A–6020 Innsbruck
joanna.opoka@uibk.ac.at

Abstract: In this paper we present a framework supporting domain-specific model assessment in a heterogeneous modelling environment. The generic method we developed is based on a common meta model for all user models created within different tools and notations. The method supports information retrieval for the set of models by means of queries and views, which can be interpreted as model cross sections corresponding to given aspects. We present a technical description of our framework: an architecture, a prototypic realisation and the technologies used in our implementation. A central point of the modular architecture is a common repository of model elements. The queries and views provide aggregated information on model elements to support the modeller during modelling and analysis processes.

1 Introduction

Nowadays there is a huge number of tools supporting model design in standardised notations such as UML¹. Together with the introduction of models in the software engineering process many novel usage scenarios of models have emerged. This ranges from model-driven architecture (MDA²) and model-based testing, e.g. [CCD04], to model-based IT-governance, e.g. [BIO05, TB06]. Not surprisingly such usage of models in real applications reveals new requirements and challenges. One of these challenges is the quality assurance of the models developed. Complex models with hundreds or thousands of model elements in general contain inconsistencies and gaps. Quality assurance of these models cannot be done by pure manual inspection or review but requires tool assistance. In the framework we are presenting in this paper we focus on the static analysis of models, which supports model developers in checking the consistency and validity of models. Our approach is based on two basic observations.

The first observation is that model quality mainly depends on the interplay of model elements and diagram types. For example, a requirements specification typically comprises a set of sub-models (like *business process model*, *use case model*, *schematic use case descriptions*) which use a set of common model elements (e.g. *action*, *use case*, *actor*,

¹Unified Modelling Language, <http://www.uml.org/>

²Model Driven Architecture, <http://www.omg.org/mda/>

business class). The use of these model elements in different models creates **complex interrelationships** and multiple sources of inconsistencies. For example, the initiating *actor* in the *use case description* has to be an *actor* related with the *use case* in the *use case model*. The model elements and their interrelationships cannot be defined in a general way for the different UML diagram types but depend on the underlying method and application context.

The second observation is that not in all cases the quality checks of a set of models can be done in an automatic way [Jug04]. An appropriate way of supporting modellers in this case is to generate model views that provide aggregated information about the models. For instance, a view on an enterprise model may list all *business classes* and related applications in table form. This information can be generated from the models and enables the quality manager to perform **cross-checks** on the model and its sub-models.

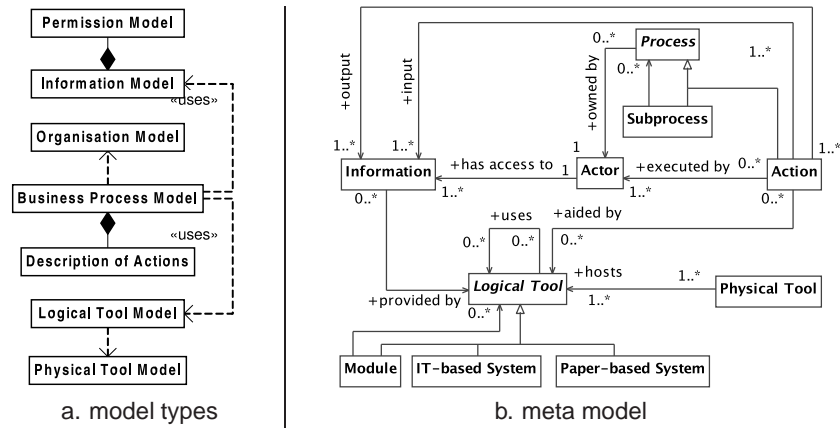
Based on these two observations we can characterise the backbone of our approach as follows. Our validation target is a set of interrelated models (which we call in the sequel a **model landscape**). We consider a heterogeneous model landscape in which the models may be UML models or other models conform to MOF³ notation. The models in the model landscape use a common set of interrelated **model elements**. The model elements together with their relationships are described in a domain-specific **meta model**. The meta model defines the universe of discourse in which quality checks can be formulated and be evaluated. Based on the structures of the meta models we define executable model validation checks. Both our meta modelling approach and the syntactic framework have been presented in [BCO05]. In this paper we focus on the **tool support** of our framework. Our tool operates on a heterogeneous model and tool landscape based on a generic **model repository**. Our method is fully generic in the sense that both the model repository and the language interpreter are independent of the meta model. Our framework can be applied in many-fold contexts in which complex model landscapes are developed. In particular, our approach is aligned with the requirements and experiences of three projects we conduct in cooperation with industrial partners. Moreover, we show that our approach is fully compliant with OMG's MDA approach.

This paper is an extended version of [COGB⁺], where we presented our tool in the context of medical process analysis. In this paper we show also other aspects and usage possibilities of our framework. The sequel of this paper is structured as follows. In section 2 the brief description of our projects and their technical and architectural requirements are presented. Section 3 includes a short summary of basic notions and concepts used in our approach. The core of the paper is section 4, in which the technical aspects of our framework are presented. Finally, the last section gives a link to related work and a conclusion.

2 Projects and Requirements

In this section we describe our cooperation projects *MedFlow*, *ProSecO*, *PRO²SA* (section 2.1) and give the summary of their requirements (section 2.2).

³Meta-Object Facility, <http://www.omg.org/mof/>

Figure 1: Model types and meta model in the *MedFlow* project.

2.1 Projects

Quality Requirement Checks Our research study on quality requirement checks was initially inspired by modelling clinical processes within the *MedFlow*⁴ project [SAWCO05]. The aim of the project is the fusion of technical and socio-organisational views of processes. In the clinical process modelling different types of models within different modelling tools and notations can be created. Such a modelling scenario, where techniques of process modelling (e.g. Aris⁵) are combined with techniques of IT-landscape modelling (e.g. using the 3LGM⁶), is typical. In Figure 1.a we depicted model types created in the *MedFlow* project (see Example 1 in section 3.1 for an exemplary instance). To be able to analyse different model types we need to define a common meta model. In Figure 1.b we depict meta model elements and relations among them. As a technical partner in the project we are interested in the technical view, where we consider IT-support of processes and we model IT-landscapes at different levels of granularity. Exemplary elements used in the analysis of IT-support are physical and logical tools defined in the Logical and the Physical Tool Model, respectively (in general a meta model element can be used in several models). For exemplary instances of meta elements see Example 1 (section 3.1) and the example in Figure 6 (section 3.2).

To analyse a heterogeneous model landscape we need a common repository to store information about all model elements. To support the modeller in the analysis of business process models we provided queries, checks and views defined on the meta model level and interpreted over a set of elements saved in the repository (see section 3).

⁴The project *MedFlow—Modelling and Assessment of Socio-Technical Processes in Health Care—A Tool-Assisted Modelling Method with Integrated code System* supported by HITT, <http://www.hitt.at/>.

⁵Aris, <http://www.ids-scheer.de/>

⁶3LGM, <http://www.3lgm2.de/>

IT Security Analysis In the research project *ProSecO*⁷ we develop a model based framework for information security management in an enterprise context [BIO05]. The approach consists of an information security model and a security management process on the basis of an enterprise model that is similar to the meta model of the *MedFlow* project (see Figure 1). We use the enterprise model to identify the relevant business and technical objects of an organisation. Every model element of the enterprise model is extended with security relevant information that is defined in the information security meta model. The extended enterprise model allows the modeller to analyse security requirements, threats and controls. To support the collaborative usage of the enterprise model by different stakeholders and to avoid inconsistencies we have defined rules that ensure the correctness and consistency of the model. The adherence to these rules is controlled running queries and checks against the extended enterprise model.

An important aspect of our model based approach to information security management are the states of the security related information that are extending the enterprise model. The security states allow the monitoring of the progress of the overall security management process. An important requirement for a supporting framework is the ability to run evaluation algorithms over the states of the security related information on either the entire or part of the enterprise model.

Strategic Alignment of Information Technology The aim of the research project *PRO²SA* is to provide adequate information for IT—and corporate—management to optimise the strategic alignment and the value delivery of IT⁸. Our approach is based on the observation that strategic information management is not properly integrated in corporate governance. A main objective of IT–governance should be the methodical integration of IT–management into enterprise–management, *narrowing the gap* between IT and business [TB06].

In one of the first steps we use a meta model oriented business engineering approach. Our business engineering domains are [TB06]: *Strategy Management, Business Development, Business Transformation* and *Information Technology*. These domains are modelled using an enterprise model that is extended with key–indicators for strategic alignment and value of IT. It is possible to define key–indicators during run–time.

The information necessary for supporting managerial decisions are extracted from UML models and integrated into the management framework of the company. For this purpose the key–indicators are analysed using evaluation algorithms that are run over the model landscape. The results are an aggregation of information over the entire set of models. The model elements stored in the repository can be analysed running queries and views and have to be visualised in different types of management charts.

⁷The project *ProSecO—A model based framework for enterprise information security management* is supported by trans IT, <http://www.transit.ac.at/>.

⁸The project *PRO²SA—Process– and Project–Oriented Strategic Alignment*

Rf_1	The tool should enable the modeller to formulate and execute queries, checks and views expressed in a predicative language based on the structure of the meta model.
Rf_2	The tool should enable the modeller to formulate and execute evaluation algorithms [PRO^2SA , $ProSecO$].
Rf_3	The tool should support meta model extensions : additional attributes for elements (at run-time) [PRO^2SA], additional state-based elements [$ProSecO$], links between elements and files with additional information [both].
Rf_4	The tool should present results in graphical and textual form (modification of model representation, graphs, trees, tables, etc.).
Rf_5	The tool should provide interfaces to different modelling tools and notations [$MedFlow$].
Rf_6	The tool should guarantee (or at least support) the consistency of models according to user-defined rules.
Rf_7	The tool should store information about a complex enterprise model landscape, where all models correspond to a given meta model and are strongly connected with each other.

Figure 2: List of functional requirements.

Rt_1	The tool should reuse and extend existing frameworks and tools.
Rt_2	The tool should be based on well-known open standards and software.
Rt_3	The tool should be platform-independent as far as possible.

Figure 3: List of technical requirements.

2.2 Common Requirements

The designed tool should meet the requirements of the three projects described in the previous sections. Below we present two kinds of requirements, namely functional and technical.

Functional Requirements The main common goal is a mechanism for aggregating information from a model landscape, which give the analyst some guidelines to further analysis (see requirement Rf_1 in Figure 2). It should also be possible to evaluate some statistics based on additional figures attached to model elements (Rf_2, Rf_3) The aggregated information should be presented using different formats (Rf_4).

In practice modellers use different modelling notations and tools, thus the second goal is to create a framework supporting consistency maintenance in heterogeneous modelling environments (Rf_5, Rf_6) with a centralised data storage (Rf_7).

Technical Requirements We decided to extend existing tools and to use established development technologies (see requirement Rt_1 in Figure 3). This requirement entails the usage of open standards and open source tools, which can be extended easier (Rt_2). To increase the number of potential users of the tool the framework should preferably be platform-independent and MDA conform (Rt_3).

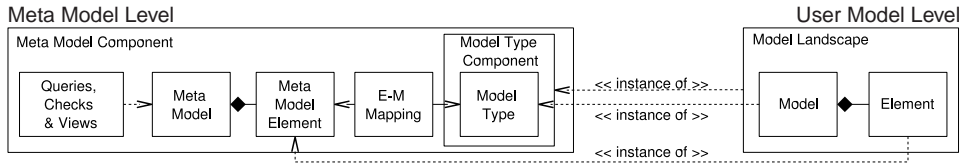


Figure 4: Modelling concepts.

3 Methodological Background

In this section we briefly describe both modelling (section 3.1) and analysis (section 3.2) within our framework. For a full description and more examples see [BCO05].

3.1 Modelling on Meta and User Levels

We consider two levels of abstraction, namely the *meta model level* and the *user model level* (see Figure 4). The *meta model* defines the model elements of the application domain (called *meta model elements*, for short *meta elements*) and the relations between them. We call instances of meta elements *elements* (see Example 1). The *model type component* defines the *model types* and their relationships. An instance of the model type component is a *model landscape* and the instances of model types are *models* (see Example 1).

Example 1 In the *MedFlow* project meta model (see Figure 1) the information element is defined. An example of its instance is $\ll\text{information}\gg$ Referral (see example in Figure 6 for other instances). An exemplary instance of Business Process Model (model type in *MedFlow* project, see Figure 1) can be a $\ll\text{BPM}\gg$ X-ray examination.

Additionally we have an element–model–mapping (an *EM–mapping* for short), which interconnects meta elements and model types. In the EM–mapping each meta element is associated with the model type where it is defined and with the set of model types where it can be used. The last concept, *queries, checks and views*, is described in section 3.2.

3.2 Model Analysis

Concepts described in the previous section create a basic environment for analysis of the model set. For analysis purpose we use different types of queries and views described in the subsequent sections.

The goal of a **query** is to provide the modeller with information on the model landscape. The queries are formulated over elements defined in the meta model. There are different types of queries regarding the type of the returned value: queries returning a boolean value

-
- *Is a given information saved in a given logical tool?* (a check, a boolean result)
 - *Number of logical tools in the model landscape.* (an integer result)
 - *The set of logical tools an information is saved in.* (a set of elements result)
 - *Each information used in BPMs has to be defined in the Information Model.*(a predefined check)
-

Figure 5: Exemplary queries for the *MedFlow* project.

-
1. The query definition: *Is a given information saved in a given logical tool?*
 2. The defining element instance sets:
 - set of all information objects defined in the model landscape:
 $I = \{Referral, Diagnostic\ Findings, Image, \dots\}$
 - set of all logical tools: $LT = \{KIS, PACS, PaterNoster, \dots\}$
 3. The result table:

<i>Information \ Logical Tool</i>	<i>KIS</i>	<i>PACS</i>	<i>PaterNoster</i>	<i>...</i>
<i>Referral</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>...</i>
<i>Diagnostic Findings</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>...</i>
<i>Image</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>...</i>
<i>...</i>	<i>...</i>	<i>...</i>	<i>...</i>	<i>...</i>
 4. The analysis of the result table:
 - A warning for the set of elements $\{Referral\}$ is activated:
Each information should have a medium!
 - A question for the set of elements $\{Diagnostic\ Findings, Image\}$ is activated:
Is the consistency of redundant information guaranteed?
 5. The further analysis of the results: a modeller could change the model to get rid of the warning and find out the answer for the question.
-

Figure 6: An example view, for *MedFlow* project, linking information objects with logical tools.

(called **checks**), queries returning an integer value, and finally queries returning a set of elements. There are two special types of queries, namely predefined checks and some complementary queries. Predefined checks are defined by the meta modeller and are well-formedness rules resulting from the structure of the meta model and the EM-mapping specific for a given application. In the example in Figure 5 we give some exemplary queries for the *MedFlow* project.

To provide more complete and cross-sectional information we aggregate the results of queries in a **view**. Views provide information on sets of elements, unlike queries, which take single instances as arguments. An exemplary view for the *MedFlow* project is given in the example in Figure 6.

4 Prototypic Architecture

The following section is concerned with the prototypic implementation of our approach. The utilised standards, tools and technologies are described in the context of our application and the underlying architecture is depicted. On the topmost level we can hereby distinguish between parts connected with meta modelling, user-level modelling, data repository

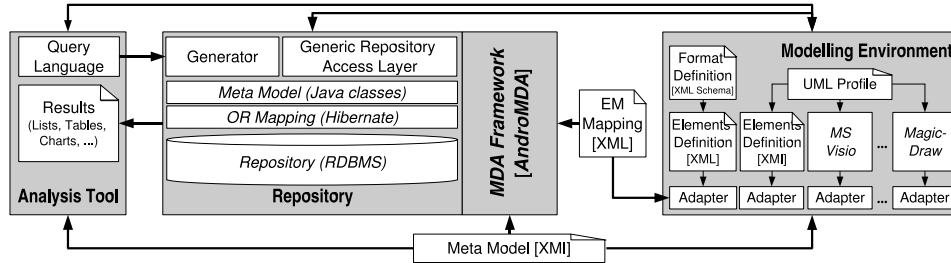


Figure 7: Architecture model of prototypic implementation.

and analysis (see Figure 7). To highlight the motivation of our design decisions references to both functional and technical requirements are listed in parentheses.

Model Repository To be able to analyse information on all model elements defined in different modelling tools (Rf_5) we decided to use one central model repository. Hereby we utilise a relational database management system to persistently store model elements generated by the different tools connected to our application. Due to the fact that our approach has to be capable of managing a user defined meta model (Rf_7) we must provide the ability to dynamically exchange both the database and the repository structure during installation time. To achieve this central requirement we rely on technologies supported by AndroMDA⁹, an open source implementation of the MDA approach (Rt_1 , Rt_2 , Rt_3). In a first step we hereby generate Java classes that reflect the structure of a MOF conform meta model provided as XMI¹⁰ file. Full support for the object/relational persistence and query service Hibernate¹¹ furthermore allows us to map these generated classes automatically to a wide range of database management systems. The choice of the RDBMS that is to be used can be postponed to the latest phase of the implementation which conforms with the requirement regarding platform independence (Rt_3). The two remaining parts of the repository (*Generic Repository Access Layer* and *Generator*) are concerned with providing meta model independent interfaces to connect both modelling and analysis tools to our common repository. Interactions with modelling tools are handled using specified XML messages sent and received via a J2EE¹² application server. The generator on the other hand will both translate queries from query languages to the repository internal representation and provide the analysis tool with aggregated results.

Modelling Environment The prototypic implementation supports two model description notations: UML and XML (Rf_5). *UML* is used in modelling tools and transformed to the repository notation with the help of adapters (separately provided for each tool). Currently we are developing adapters for *MS Visio* and *MagicDraw* (Rf_5). The former is

⁹AndroMDA, <http://www.andromda.org/>

¹⁰XMI, <http://www.omg.org/technology/documents/formal/xmi.htm>

¹¹Hibernate, <http://www.hibernate.org/>

¹²Java 2 Enterprise Edition, <http://java.sun.com/j2ee/>

not a strict UML modelling tool but as it is widely-used in practice providing support for it is essential for our approach. MagicDraw on the other hand is a sophisticated platform-independent UML modelling tool (Rt_3). The tool-independent adapter is concerned with models described using XML (Rt_3) in a predefined format. To assure the repository conformity of models intended to be imported into the model repository all adapters interpret the EM-mapping file and additionally derive class meta-information from the AndromDA generated Java classes using Java reflection mechanism (Rf_6).

Analysis Tool Using the query language modellers can construct their own queries and views (Rf_1 , Rf_2) to aggregate information about models contained within the repository. Both HTML pages and images illustrating the results graphically (Rf_4) will be generated. As our tool is implemented by means of server-side J2EE technologies network based access can easily be provided.

5 Related Work and Conclusion

In this section we compare our approach with similar solutions, we stress a novelty of our framework and give examples of different types of tools that can be used for model analysis. In the last paragraph we conclude our results.

Existing methods and tools focus mainly on syntactical correctness independent of modelling domains, as well as on providing modelling metrics [Gro04, Jug04]. The proposed indicators of high quality models, such as for example general diagramming or diagram-specific metrics, are too general and domain independent. Our approach focuses on **semantic, domain-specific analysis** based on the conceptual model of a given domain, which allows us to define checks over an entire model landscape. There are several approaches, mostly in industrial contexts that deal with quality checks of model landscapes. Developed tools are dedicated for specific applications and support fixed sets of checks, e.g. in [HMT02] a meta model for describing IT landscapes has been defined. In our approach **different meta models** can be used and modellers can define their **own queries**.

Existing approaches deal with homogeneous modelling environments. In a study at BMW [Jug04] a set of quality checks for model landscapes has been developed. The checks are input to an in-house MDA tool. As further example Siemens Princeton has developed a tool¹³ for checking requirements specifications. For the implementation these approaches use the scripting facilities or programming interfaces of UML tools or graphic programs such as Rational Rose, Together, Visio or Adonis¹⁴. Our approach is dedicated to a **heterogeneous modelling environment** and the queries are defined and interpreted in a tool independent way, only adapters are connected with specific tools.

We also consider **heterogeneous modelling notations**, while existing tools are mostly dedicated to one notation only. For example the Executable UML [MB02] can be used to

¹³Design Advisor tool, <http://www.scr.siemens.com/>

¹⁴Business process management toolkit, <http://www.boc-eu.com/advisor/adonis.html>

execute and test models defined in UML. And with the OCLE tool¹⁵ it is possible to check UML models against well formedness rules, methodological, profile or target implementation language rules expressed in OCL. It is also possible to obtain metric information about UML models. This tool also uses XML, but only as a data tier, while we use XML also for modelling purposes.

This paper presents a technical description of our framework for a domain-specific model assessment. Our methodology is based on a common meta model describing model elements and their relationships. All models in the model landscape are meta model conform, thus model elements can be saved in a common repository. Using queries and views the information saved in the repository can be analysed. We showed how to implement our methodology and integrate it with heterogeneous modelling environments. The implementation is work in progress. There are some limitations in the prototypic version of our tool, such as a limited number of adapters or hard coded queries. In further work we plan to implement a generic analysis tool and to increase the number of supported modelling tools also for non-UML (but MOF conform) notations. Moreover run-time meta model extension (Rf_3) are as well subject of future tool extensions.

References

- [BCO05] R. Breu and J. Chimiak-Opoka. Towards Systematic Model Assessment. In *ER 2005 Workshops*, volume 3770 of *LNCS*, pages 398–409. Springer, 2005.
- [BIO05] R. Breu and F. Innerhofer-Oberperfler. Model based business driven IT security analysis. In *Proc. SREIS*, August 2005.
- [CCD04] A. Cavarra, C. Crichton, and J. Davies. A method for the automatic generation of test suites from object models. *Inf. & Software Tech.*, 46(5):309–314, 2004.
- [COGB⁺] J. Chimiak-Opoka, G. Giesinger, R. Breu, S. Saboor, and E. Ammenwerth. Tool-Supported Analysis of Clinical Processes. In *Biomed 2006*, pages 56–59. ACTA Press.
- [Gro04] R.C. Gronback. Model Validation: Applying Audits and Metrics to UML Models. *Borland Developer Conference*, 2004.
- [HMT02] M. Heberling, C. Maier, and T. Tensi. Visual Modeling and Managing the Software Architecture Landscape in a Large Enterprise by an Extension of the UML. In *Second Workshop on Domain-Specific Visual Languages. An OOPSLA Workshop*, Nov 2002.
- [Jug04] F. Jug. Methods and techniques for quality assurance in software development process in BMW group (in German). Master’s thesis, Technical Univ. Munich, Sep. 2004.
- [MB02] S.J. Mellor and M.J. Balcer. *Executable UML. A Foundation for Model-Driven Architecture*. Addison-Wesley, 2002.
- [SAWCO05] S. Saboor, E. Ammenwerth, M. Wurz, and J. Chimiak-Opoka. MedFlow—improving modelling and assessment of clinical processes. In *Proc. MIE*, pages 521–526, 2005.
- [TB06] B. Tilg and R. Breu. PROSA—Ein modellgetriebener Ansatz zur Integration des Strategic Business Alignments in die Business Intelligence. MKWI (accepted), 2006.

¹⁵Object Constraint Language Environment, <http://lci.cs.ubbcluj.ro/ocle/>