Budapest University of Technology and Economics
Department of Automation and Applied Informatics

TRANSFORMATION-BASED SUPPORT FOR VISUAL LANGUAGES

Ph.D. Thesis

Gergely Mezei

Advisors:
Dr. Hassan Charaf, Ph.D.
Associate Professor

Dr. Tihamér Levendovszky, Ph.D.
Senior Lecturer

BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS
DEPARTMENT OF AUTOMATION AND APPLIED INFORMATICS

Budapest, 2007

Ph.D. Thesis

Gergely Mezei

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Automation and Applied Informatics

1111 Budapest, Goldmann György tér 3.

e-mail: gmezei@aut.bme.hu
tel: +36(1)4631007
fax: +36(1)4633478

Advisors:
Dr. Hassan Charaf, Ph.D.
Associate Professor

Dr. Tihamér Levendovszky, Ph.D.
Senior Lecturer

1 Introduction and Motivation

With the increasing complexity of software systems, the model-based software development became popular. Models are capable of describing the problems and their solutions on a high level of abstraction. This abstraction ensures that important aspects of the problem set can be focused, while non-important aspects remain hidden. The high level of abstraction, and the visual representation of models can definitely help in solving complex engineering tasks and problems.

It is possible to create more, than one model for a problem based on the abstraction level, on the modeling language and on the paradigms. This means that it is essential to have models that are self-evident and precisely defined. The Unified Modeling Language (UML) [UML], which is one of the most popular modeling languages, offers a solution to these problems. UML is a visual language family defined by formal syntax. By using UML, it is possible to illustrate, solve the problems and document the solutions of software development.

UML offers the same tool set regardless of the modeled problems. This standard in modeling tool set increases the understandability of models, however, it can result in inflexible models that are completely missing the specialities of the target domain. Due to the inflexibility of UML, the popularity of Domain-Specific Modeling Languages (DSMLs) [DSML] has increased. The main idea behind the domain-specific modeling is that it defines a language (or language family) for each domain of interest. These languages can support the specialities of the domain, but their usability is heavily restricted. By supporting a domain-specific tool set and notation, DSMLs simplify the specialized illustration of problems even for non-IT experts, which improves efficiency both modeling and the model-based approaches.

Besides the advantages of domain-specific modeling languages, it is a usual problem to define these languages because of their flexibility. Similarly, it is problematic to create a tool that supports DSMs, because of the large differences in the target domains. However, these issues can be solved by metamodeling [Sztipanovits, 1997]. Metamodeling is an efficient technique to define visual languages including domain-specific languages. Simply said, a metamodel is the model of a model. The metamodel is a paradigm, a set of rules that the modeling environment should keep during the modeling. Metamodels define the modeling types, the relations between them and the attributes of both the types and the relations. Once a metamodel is defined, instances of this metamodel, more precisely models conforming to the metamodel rules can be created.

This type-instance relation is not restricted to two modeling layers, but it can be extended to further modeliny layers. For example, we can construct meta-metamodels, which define rules for the metamodel. Higher modeling layers mean obviously higher abstraction level as well, in other words, a metamodel is more abstract, than its model. One of the most well-accepted metamodeling standard is the Meta-Object Facility (MOF) [MOF]. Originally, MOF has defined a four-layer modeling hierarchy. However, the depth of the modeling hierarchy is theoretically not limited. By increasing the number of modeling layers, we can reduce the gap between the abstraction level of models and their instantiation, thus, we can refine the modeling hierarchy. However, allowing n-layer metamodeling hierarchies can lead to several problems. One part of these problems are topological, other part is mainly related to the attribute instantiations. The topology-related problems have already been solved in theory and in practice as well [Levendovszky, 2005]. Nevertheless, the issues of n-layer attribute instantiation is not solved.

Metamodeling can describe the topological rules of an arbitrary domain. However, metamodeling is not meant to describe the domain-specific visualization as well. Therefore, metamodeling requires a method to describe the notation of visual language elements.

Visual languages have the tendency to create incomplete, informal, imprecise, and sometimes even inconsistent models. These issues apply to UML and to metamodeled languages as well. The source of the problem is that there can be complex relationships between the model items, and these relationships are hard (or even impossible) to solve with visual definitions. One of the most well-accepted solution to this problem is to attach textual constraints to the model items. Practice has shown that using natural languages to describe these constraints results in ambiguities. Object Constraint Language (OCL) [OCL] is a formal, textual and declarative language devoted to help the creation of precise models. OCL is possibly the most popular, or at least the most widely used, constraint language in modeling. Initially, OCL was only a formal specification language extension to UML. Later, it has been used with any MOF metamodel, including but not restricted to UML. OCL is a quite flexible language. However, the flexibility of OCL makes it capable of describing constraints in an n-layer metamodeling system.

Since we need precise models, the evaluation of constraints is also important. Besides the flexibility of OCL, the performance of constraint evaluation is also important due to the wide range of usage. During the optimization, the constraints are re-defined by keeping the original semantics, but decreasing the time required by the evaluation. Since the constraints are changed, the equivalence of the original and the new constraint must be proven.

In modeling, model processing techniques are essential, unless we use models only for documentation purposes. Model-Driven Architecture (MDA) [Soley et al., 2004] provides guidelines to structure specifications expressed as models. MDA separates the design from the architecture. More precisely, it defines a platform-independent model (PIM), a Platform Definition Model (PDM) and several platform specific models (PSMs). PIM is usually created by using UML and other modeling languages capable of generating platform-dependent artifacts automatically by model compilers. Model-Integrated Computing (MIC) [MIC] is another model-based approach. MIC uses domain-specific models to synthesize applications. The key element is the extension of the role of models such that they form the „backbone" of a model-integrated system development process. In order to achieve this, MIC focuses on models, modeling environments and facilitates code generation from the models. Metamodeling environments and model interpreters together form the tool support for MIC. Model transformation plays an important role in both cases and in another model-driven approaches as well.

Nowadays, models are often represented by labeled, directed graphs. This description makes it possible to use graph-rewriting in model transformation. The rules used in rewriting consist of a left-hand side (LHS) and a right-hand side (RHS). When applying the rule, we are trying to find the pattern defined in the LHS, and replace it with the pattern defined in the RHS.

Although graph-rewriting is an efficient method in case of small models and patterns, the pattern matching algorithm used in the rewriting is an NP-complete problem. This means, however, that increasing the size of patterns would result in slow transformations. Since the complexity of matching cannot be reduced in the general case, we need to increase the computation capabilities. One possible way to solve this issue is to use parallelism. In the general case, the dependency between the rules can limit the level of parallelism, but it is worth to use parallelism in other cases. Parallelism can be used

either at the transformation-level, or in pattern matching.

The main topic of the presented research is the transformation-based support for visual languages. The open issues related to the efficient model transformation are as follows:

1 **N-layer attribute structure.** To support n-layer metamodeling, an attribute structure should be created. This structure must be layer transparent and flexible enough to describe the attributes of domain-specific languages. The attribute structure must be formalized. A precisely defined, layer-transparent instantiation transformation is also needed.

2 **Supporting appearance definitions.** Metamodeling does not support defining the appearance of visual models. A new method is needed that fits the metamodeling concepts and is able to describe the notation of the language. A model processing technique is also needed to map and bind the appearance definitions to the topological definitions.

3 **Constraints extensions.** In order to make the constraint validation possible in n-layer metamodeling environments, an extension, a new dialect must be provided to one of the existing constraint languages. The new dialect should be formalized.

4 **Optimized constraint evaluation.** There is a need for an efficient constraint handling solution. To achieve this, the optimization of OCL constraints must be supported. Moreover, the correctness of the optimization algorithms must be proven in a formal way.

5 **Parallel model transformation.** The performance of graph rewriting-based model transformations must be improved by the parallel execution of the transformations. Examination is needed to explore the parallelization possibilities at the transformation level and at the level of rewriting rules. Distributed algorithms must be created and the efficiency of these algorithms must be proven by performance measurements.

2 Methodological Summary

The requirements above determined the directions of my research and the individual tasks that should be solved one after another. The starting points of my work were the already existing modeling approaches, modeling tools, the constraint management mechanisms, and the existing model transformation solutions.

As an introduction to modeling, I have studied algorithms and theorems and of **graph theory**. This information was useful in handling graph-based modeling. In the next step, I read the specification of MOF. In my research, I used the propositions of [Levendovszky, 2005] in handling n-layer topological rules. These results are extended by my attribute structure and attribute instantiation mechanism. In formalizing the structure and the instantiation, I used Abstract State Machines (ASMs) [ASM].

In creating a method to define the appearance of metamodeled items, I used a graph rewriting-based model transformation. In analyzing the properties of this transformation, I used the constructs of **category theory** [Pierce, 1991].

Dealing with metamodel-based constraint handling, I studied the existing formalism of OCL [OCLFormalism]. This formalism is based on set theory, thus, I studied the basics of

set theory as well. Extending the formalism, I used the aforementioned n-layer attribute structure. Applying the constraint optimization, I borrowed a few ideas from aspect-oriented constraint handling [Lengyel, 2006]. The refined version of these ideas were used in the first optimization algorithm. When creating the second optimization algorithm, I studied the optimization possibilities for generic compilers [Atho et al., 1998]. Proving the correctness of the optimization algorithms, I used constructs of **automata theory** and the ASMs.

The theoretical background of parallel model transformations were the mathematical background of graph transformations [Rozenberg, 1997], and the Double Pushout (DPO) approach. Graph transformations used in my approach are strictly ordered. This strict order means that the transformation defines not only the rewriting steps themselves, but their order of execution as well. The rules of the transformation are metamodel-based, thus, both the LHS and the RHS are built from metamodel items. This also means that we are searching for an isomorph subgraph with an instantiation of the LHS and not an isomorph subgraph of the LHS itself.

Furthermore, selecting the rules applicable in parallel, I used the Parallelism theorem of the DPO approach, more precisely an extension of the theorem [Levendovszky, 2005] that is applicable for metamodel-based rewriting rules. Parallelizing the matching, I borrowed the concept of dynamic load balancing from [Kumar et al., 1994].

At the Department of Automation and Applied Informatics of BUTE we developed the Visual Modeling and Transformation System (VMTS) [VMTS] that is a metamodeling and model transformation framework. During my research, I have implemented the theoretical solutions in modules of VMTS in order to show the practical relevance of the solutions as well. Moreover, based on the feedback provided by the experiences, I continuously improved the theoretical background of my research.

3 Novel Scientific Results

The results that I have contributed in this work are summarized in three theses. I have proven these results with engineering and mathematical methods, and I have illustrated their practical relevance in engineering applications.

The first thesis deals with the definition of visual languages. The attributes of meta-modeled visual languages are focused. A new attribute structure is provided, which is defined with a formal syntax. The structure is highly flexible and it can be used on any metamodeling layer. An instantiation transformation is also elaborated for the attribute structure. The appearance definition of visual languages is focused. We provide a method that can model the appearance by using a special visual language and map appearance definitions to topological definition by model transformation.

The second thesis deals with the constraints of visual languages. In order to define precise visual languages, the language definition is often extended by constraints. The contribution extends one of the most popular constraint languages, OCL, to support the n-layer attribute structure. Moreover, optimization methods are provided for OCL-based constraint optimization. The aim of the algorithms is to reduce the number of model queries by relocating the constraint, or by applying a special caching method during the constraint evaluation.

The third thesis focuses the transformation of visual languages. We present a new model transformation approach that supports executing transformations in parallel.

Firstly, the transformation-level parallelism is focused, where the transformation rules are handled as atomic units. Secondly, the rule-level parallelism is introduced, which uses parallelism during the matching process. The parallelism approaches are unified and their performance is analyzed by measurements.

The theoretical results of my research have been validated in practice by implementing the results. In the booklet, this validation is also elaborated.

Thesis I
The Definition of Visual Languages

To support the definition of visual languages, I have provided an attribute structure and an attribute instantiation transformation that is applicable in n-layer metamodeling.

Thesis I is contained by Chapter 4 of the dissertation. Related publications: [4][6][9][10][11][17][18][19][20][32][34][37][44].

I have created a new, generic attribute structure. The formal definition of the structure grants the preciseness, while the generic constructions of the structure grants the expressiveness.

Definition 3.1. *The superuniverse* $|\mathfrak{A}|$ *of a state* $\mathfrak{A}$ *of the Simplified N-Layer Attribute Algebra is the union of four universes:*

- U_{Bool} *containing logical values {true/false}*

- U_{Number} *containing rational numbers* $\{\mathbb{Q}\}$

- U_{String} *containing character sequences of finite length*

- U_{ID} *containing all the possible IDs.*

Definition 3.2. *The vocabulary* Σ *of the Simplified N-Layer Attribute Algebra formalism is assumed to contain the following characteristic functions (arities follow the slashes): Meta/1, IsAttribute/1, Name/1, Root/1, Attribute/3.*

Subthesis I.1

I have introduced an attribute structure following the definition above. I have proven that the algebra can describe n-layer attribute structures. I have created mathematical formulas to define valid models.

I have shown that it is possible to create a transformation algorithm which can create the attribute structure of instantiated models items for each valid model item defined by the algebra. I have formalized the transformation algorithm.

I have proven that the presented algebra and the instantiation algorithm together can model the attribute structure of UML 2.0 compliant class diagrams, object diagrams and statechart diagrams.

Definition 3.3. *In a modeling hierarchy, an instantiation is called deep instantiation if the information, which is modeled on the n. modeling layer, is available on the n+x. layer* $(x > 0)$ *as well [Atkinson and Kuhne, 2001].*

Since the support for deep instantiation is a common request in metamodeling, I have extended the Simplified N-Layer Attribute Algebra in order to support deep instantiation as well.

Definition 3.4. *The vocabulary Σ of the Enhanced N-Layer Attribute Algebra formalism is assumed to contain the following characteristic functions (arities follow the slashes): Meta/1, IsAttribute/1, Name/1, Root/1, Attribute/3, ReInstantiate/1, ReInstantiation-Name/1 .*

Subthesis I.2

I have shown that it is possible to construct an attribute algebra that supports n-layer attributes and deep instantiation as well. The superuniverse of the algebra equals the superuniverse of Simplified N-Layer Attribute Algebra, while the vocabulary of the algebra follows the definition above.

I have verified that it is possible to create a formal algorithm for the new algebra, which can execute instantiation.

I have introduced semi-free instantiations of models, which means instantiating the model and modifying the result model by not breaking the validity rules. I have proven that the validity of the original model grants that the n-ary self-embedding of the semi-free instantiation function exists and it produces valid models.

I have proven that applying the instantiation transformation on models following the Enhanced N-Layer Attribute Algebra always terminates.

I have shown that the new algebra and the instantiation algorithm together can generate the attribute structure of UML 2.0 object diagrams based on UML 2.0 class diagrams.

I have verified that the n-layer attribute structure and instantiation algorithm can be created in practice. As reference implementation, I have created an XML-based and an object-oriented implementation. The applicability of the attribute has been proven in industrial projects as well, e.g. in case of feature modeling.

I have given a method to model the appearance definition of visual languages and a method to map the appearance definition to the topological definition provided by the metamodel. I have defined a domain-specific language, the VMTS Presentation DSL (VPD) by metamodeling. The VPD language is able to describe the visualization of other domains. I have created a theoretical tool set that defines the appearance by composing simple elements.

Subthesis I.3

I have proven that it is possible to create a metamodel that defines the VPD language. Furthermore, I have shown that the VPD model describing the appearance can be mapped to the topological, structural rules of the target domain defined by the metamodel (Fig. 1).

I have shown that the mapping between the appearance and the topological definition can be automated by model transformation. I have also shown that this model transformation can be described by a sequence of graph rewriting rules, and I have created these rules.

I have verified that the VPD language is able to define the visualization of UML 2.0 compliant activity diagrams, statechart diagrams, flowchart diagrams and the Nessie-Schneidermann diagram.

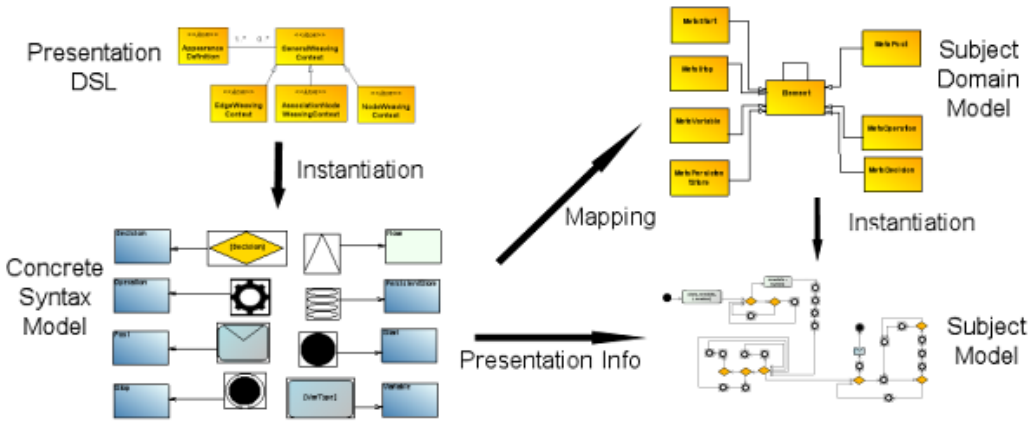

Figure 1: Mapping the appearance to the topological information

Using a model transformation to process the concrete syntax definition is a straightforward solution, because the changes in the modeling methods or in the modeling structure can be easily adapted. This flexibility has some drawbacks: if the transformation changes, its correctness must be proven again. One of the most important properties is to decide whether the transformation terminates. I used the definitions and theorems presented in [Levendovszky, 2006] to make the proof method simpler.

Subthesis I.4

Using the E-based composition of graph transformations, I have proven that the model transformation binding the appearance and topological information always terminates, when the input graph (the topological model) is finite.

Thesis II
Optimized, Metamodel-Based Constraint Validation

To support the efficient handling of precise and complete modeling languages, I have supplied an extension that supports the n-layer attribute algebra for OCL. Furthermore, I have given two optimization algorithms to reduce the time of constraint evaluation. The first algorithm focuses on reducing the number of model queries by relocating the constraint, while the second algorithm uses the specialities of OCL to create an efficient caching mechanism. Moreover, I have given a new formalism for OCL that supports both the attribute extensions and modeling the evaluation of constraints. The new formalism was used in proving the correctness of the optimization algorithms.

Thesis II is contained by Chapter 5 of the dissertation. Related publications: [2][5][7][8][21][26][31][35][36][39][46][47][48].

OCL is a formal language, its formalization is based on set theory [OCL Spec., 2003]. However, the existing formalism [OCLFormalism] does not cover all features of OCL, and it does not define the evaluation of constraints. Furthermore, set theory is a flexible technique, but it uses a low-level description of the problem space, and it does not support modularization. I have created the OCLASM as a new formalism for OCL. The basic concepts of OCLASM are borrowed from [Stark et al., 2001].

OCLASM acts as an interpreter for OCL constraints, it describes the execution of constraint expressions. The states of OCLASM can be considered as internal state of the evaluating environment, which is evolved by the rule set of OCLASM. Functions in this case are used to query the underlying model, obtain the expressions from the constraint and to represent the internal state of the interpreter simulated by OCLASM. The formalism uses monitored functions to query the model items. The constraint expressions are defined by shared functions to ensuring that these expressions can be set be external users and by internal algorithms (e.g. optimization algorithms). OCLASM creates a syntax tree from constraint expressions, this tree is processed during the evaluation.

Definition 3.5. *The vocabulary OCLASM of the OCLASM formalism is assumed to contain the following characteristic functions (arities follow the slashes):*

Monitored functions: *IsModelItem/1, AttrValue/1, Meta/1, To/2, Mul/2*

Shared functions: *GetPhrase/1, Child/2, Parent/1*

Dynamic functions: *SelfReference/0, CurrentPos/0, Type/1, Value/1, Name/1, Local/1*

Definition 3.6. *The superuniverse $|\mathfrak{A}|$ of a state $\mathfrak{A}$ of OCLASM is the union of six universes: (i) The universe of Phrases (basic syntactic constructs of OCL as defined in the EBNF definition of OCL [OCL]) ($U_{Phrases}$) (ii) The universe of possible tree positions of Phrases in the constraints (U_{Pos}) (iii) The universe Boolean ($true/false/undef$, U_{Bool}) (iv) The universe of finite lists of numbers (U_{Number}) (v) The universe of finite lists of finite strings (U_{String}) (vi) The universe of finite lists of possible identifiers (IDs) for the model items and attributes (U_{ID}).*

Subthesis II.1

I have shown that OCL can be extended to support n-layer metamodeled attributes following the attribute algebra provided in Thesis I. I have provided an OCL dialect containing this extension.

I have supplied OCLASM, a new formalism of OCL. I have shown that it is capable of describing the aforementioned OCL dialect and that it supports describing the dynamic behavior of constraints during the evaluation.

One of the most efficient way to accelerate the constraint evaluation is to reduce the navigation steps in a constraint without changing the result of the constraint. This is the aim of the first algorithm called *RelocateConstraint*. The algorithm tries to find the „optimal" model item for the constraint, where optimal means that the evaluation of the constraint needs the least possible model queries. During the evolution of the algorithm, I used the results of previous research [Lengyel, 2006]. I have improved these results, and extended them by supporting multiplicities. This extension was necessary in order to use the basic ideas in case of modeling constraints.

Subthesis II.2

I have proposed an algorithm that can reduce the number of queries based on constraint relocation. The path between the original and the new context is referred to as Relocation-Path. I have shown that stepwise application of the relocation along this path does not

restrict the relocation possibilities, thus, all valid relocation can be modeled by a sequence of relocation steps between direct neighbors.

I have shown that a constraint must be reformulated in order to use it in the new context. I have proven that the multiplicities of the underlying metamodel can affect this reformulation. I have elaborated all possible multiplicity combinations between two model items, and I have proposed constructs to handle the different cases. I have proven that the relocation is not possible if the destination side allows zero multiplicity, but it is always possible in any other cases.

I have given a formal definition of the relocation algorithm in OCLASM, and I have proven its correctness formally.

The second algorithm (*ReferenceCaching*) is based on the fact that OCL constraints cannot change the underlying model. The optimization algorithm (the *ReferenceCaching* algorithm) has two main steps: (i) obtaining statistical information about the model references (*GetCommonReferences* algorithm), and (ii) caching the evaluation expressions (*CachingManagement* algorithm). *ReferenceCaching* realizes a constraint caching method with offline control flow analysis-based reference counting to avoid unnecessary cached attributes.

Definition 3.7. *In compiler optimization, two references to the same variable or field is called common subexpression [Atho et al., 1998], if the value of the variable or field does not change between the references.*

Subthesis II.3

I have shown that the constraint expressions are always „common subexpressions", namely the result of the first model query always equals to the result of later queries.

I have created the GetCommonReferences algorithm, and I have proven that the algorithm can be applied offline.

I have provided the ReferenceCaching algorithm. I have shown that the number of applied model queries cannot increase by applying the algorithm. Moreover, I have proven that the algorithm does not store model itmes, which are referenced more than once.

Using the OCLASM formalism, I have proven that the ReferenceCaching algorithm is always correct.

I have verified that it is possible to create an OCL compiler in practice that (i) supports the new OCL dialect, (ii) supports relocation and (iii) supports constraint caching.

I have proven by performance measurements that the optimized constraints are more efficient, than the original constraints.

Thesis III
Truly Parallel Graph Transformation

Usually, a graph-rewriting based model transformation consists of several rewriting rules. The complexity of a single rewriting step is $O(n^k)$, where n is the size of the input graph, and k is the size of the pattern defined in the rewriting rule. There exist several approaches which can reduce this complexity in special cases, but these solutions are not applicable in general. Thus, the complexity of the whole transformation can be expressed

as $O(\sum_{i=1}^{r} n_i^k)$, where r is the number of transformation steps. This complexity can be reduced by applying transformation steps in parallel. However, the parallelization is limited by the dependencies between the transformation steps. It is possible to use heuristic algorithms in order to avoid or eliminate this conflicting dependencies.

Parallel execution can take place not only on the transformation level, but on the rule level as well. Rule level means in this case that the rewriting of a single rule is applied in parallel. Pattern matching in rewriting can be accelerated by starting the matching from different starting points. However, it is possible that we search only for the first match, not for all matches, thus parallel execution is not significantly faster than the sequential algorithm.

In my research, I used strictly ordered model transformations that are based on metamodel-based rewriting rules following the DPO approach. In this case, I have shown that is possible to increase the performance of model transformations, if parallelism is used. I have shown that the efficiency of parallelism is affected by the properties of the transformation. I have created two transformation-level and two rule-level parallel algorithms. I have proven that the result of the algorithm is always correct, and I have analyzed the efficiency of the algorithms in practice. Moreover, I have created a unified framework that supports applying the different levels of parallelism in an unified way.

Thesis III is contained by Chapter 6 of the dissertation. Related publications: [38][40][42][50].

I have defined the $\mathbb{I}$, $\mathbb{P}$ és $\mathbb{T}$ relations, which obtains the inherited types, base types and the metamodel of the selected model item.

Definition 3.8. *In a metamodel-based transformation, two rewriting rules r_1 and r_2 are in metaconflict, if r_1 deletes, or creates a metamodel element X, for which $\mathbb{I}(X) \cap \mathbb{S}(L_2) \neq \{\varnothing\}$ is not empty, where L_2 is the LHS definition of r_2, or vica versa r_2 deletes or creates a metamodel element, for which $\mathbb{I}(X) \cap \mathbb{S}(L_1) \neq \{\varnothing\}$ (L_1 is the metamodel definition of r_1).*

Definition 3.9. *A rule in an independence block is always parallel and sequential independent from any other rule in the block.*

In order to obtain independence blocks, I collect rules that were adjacent rules in the original transformation and which are not in metaconflict with each other. Before applying the transformation, it is converted to a sequence of independence blocks.

Subthesis III.1

I have shown that the existence of metaconflict relation between two rules can be applied by an offline algorithm.

I have proven that all transformation can be divided into a sequence of independence blocks. I have shown that the construction of these blocks can based on metaconflict examination. I have given an offline algorithm to construct the blocks.

I have given a set of algorithms that together allow executing independence blocks overlapped. I have proven that the execution is correct even if the rules are dependent on each other.

Since in case of overlapped execution conflicts may occur between the rules, thus, I have supplied heuristic algorithms to avoid the conflicts, and an algorithm to resolve conflicts, if they still occur. I have proven that this modified algorithm can also be used in transformation-level parallelism methods.

Subthesis III.2

I have shown that in case of metamodel-based rewriting rules, the host model can be partitioned along the metamodel items of the rule, and the size of metamodel partitions affect the time of matching. I have given an exact, refined formula for the cost of matching:

$$\prod_{i=1}^{k} \big(S(rn_i) - \sum_{j=1}^{i-1} N(rn_i, rn_j)\big), \tag{3.1}$$

where k is the number of nodes in the selected rule, rn denotes the nodes in the rule, and N is a function defined as:

$$N(l,m) = \begin{cases} 1, & if\ \mathbb{T}(m) \in \mathbb{I}(\mathbb{T}(l)), \\ 0, & otherwise \end{cases}$$

I have provided a rule-level parallel algorithm. Rule-level means that matching of a single rule is computed in parallel. The algorithm selects a (meta)node in the rule definition and creates a list of possible pairs (pivot points) in the host model. The elements of this list are computed in parallel. I have shown that the complexity of matching is

$$\prod_{i=2}^{k} \big(S(rn_i) - \sum_{j=2}^{i-1} N(rn_i, rn_j)\big) * S(rn_1)/c * h, \tag{3.2}$$

where k is the number of nodes in the rule, rn_1 is the selected rule node, c is the number of clients and h is the quotient of calculating a match locally and sending/receiving a match via the network defined by its pivot point.

I have given another rule-level parallel algorithm that randomizes the order of evaluation for the pivot points (not restricted to the firstly selected rule node). I have shown that all clients evaluate all possible pivot points, but in possibly different order.

The main drawback of transformation-level parallelism is that it cannot handle (cannot parallelize) the major differences in rewriting times. However, the rule-level parallelism offers a method to reduce the time of matching. By composing the two approaches, we can obtain a truly-parallel transformation engine that benefits from the advantages of both approaches by supporting both levels of parallelism.

Subthesis III.3

I have shown that it is possible to unify the transformation-level and rule-level parallelism algorithms and construct a truly parallel transformation approach. The architecture of the unified approach consists of two levels: the clients of the first level are responsible for applying transformation-level parallelism, while the client of the second level use rule-level parallelism.

I have shown that it is possible to bind the second-level clients dynamically to the first-level clients. This dynamic binding improves performance.

I have proven that the unified framework can be constructed in practice as well. I have analyzed the performance of parallel execution by a concrete case study and measurements. According to the results, I have shown that the main weakness of transformation-level parallelism is that it cannot apply the transformations efficiently in parallel, which contain rules requiring considerably different times to match. However, this issue is solved by rule-level parallelism.

4 Applications of the Novel Scientific Results

The new scientific results are implemented in the Visual Modeling and Transformation System (VMTS) and its applications. VMTS is an n-layer metamodeling and model transformation environment. VMTS uses a simplified UML class diagram as its root metamodel („visual vocabulary"). VMTS is an approach that uniformly treats model storage and model transformation. Moreover, VMTS has built-in support for several visual languages including UML diagrams, feature models, and also provides facilities to create custom domain-specific languages simply. In VMTS, precise models can be created by defining OCL constraints on metamodel items. VMTS supports two kinds of model transformations: a traversal approach and a graph rewriting-based approach.

The case studies implemented in VMTS make it possible to show the practical relevance of the presented results both for the scientific and for the industrial world. Since the evolution of the theoretical and practical background is continuous, thus, the list of applications is not final.

Related publications:
[1][3][12][13][15][14][16][22][23][24][25][27][28][29][30][33][41][43][45][49].

Framework

To prove the practical applicability of the results, I have developed the following components of the VMTS (Fig. 2): (i) VMTS Presentation Framework that is the base of the modeler application (Adaptive Modeler). The Presentation Framework offers a user-friendly way to create and edit visual languages, moreover, it uses a plug-in-based architecture to implement special domain-specific languages. (ii) Attribute Panel that supports processing and visualization of the attributes and the automatic validation of the attribute scheme based on the metamodel definition. (iii) OCL Compiler that compiles textual constraints attached to the model items and creates a validating binary from them. (iv) AGSI Compact Framework that supports the memory-based representation and the serialization of model items. (v) AGSI Parallel Transformation Engine that is the unified parallel model transformation framework. Other parts of VMTS were implemented by Tihamér Levendovszky, László Lengyel and numerous students.

Applications

Modeling and transforming of visual languages have been used in several practical cases to solve engineering problems. Using the attribute structure, the modeler framework and the optimized OCL constraints, several domains of interest became simpler to be

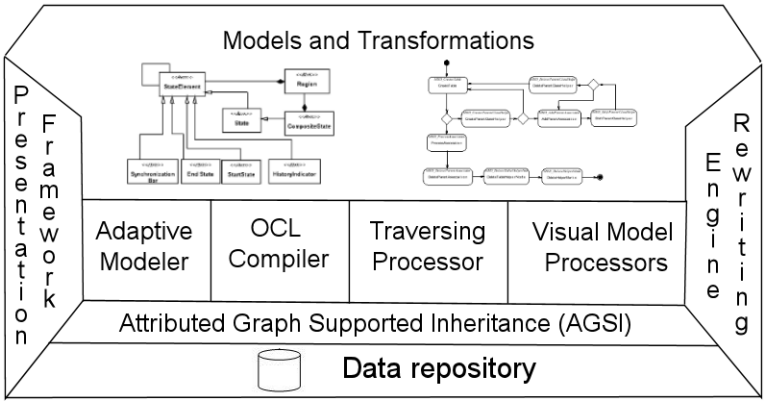

Figure 2: Architectural overview of VMTS

model: (i) Visual modeling languages to define model transformation (VMTS Control Flow Language) and rewriting rules [3][12]. (ii) Graphical user interface editors for mobile phones [13][22]. (iii) Support for UML 2.0 compliant models including but not restricted to activity-, object-, class-, sequence-, and statechart diagrams. (iv) Visual modeling language to define the appearance of domain-specific languages [4][18]. (v) Support for visual programming languages (Flowchart diagram and Nessie-Schneidermann diagram). (vi) Modeling mobile network protocols.

Using the memory-based AGSI Compact Framework and parallel model transformations, it became possible to process more complex models. The new solutions are about two magnitude faster.

The sample transformations illustrate that VMTS is an efficient model transformation tool, defining, editing and processing models.

5 Related Publications

Journals

1 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, "A Strict Control Flow Specification for Model Transformation", WSEAS Transactions on Computers, February 2006, ISSN 1109-2750.

2 **G. Mezei**, L. Lengyel, T. Levendovszky, H. Charaf, "Minimizing the Traversing Steps in the Code Generated by OCL 2.0 Compilers", WSEAS Transactions on Information Science and Applications, Issue 4, Volume 3, February 2006, ISSN 1109-0832, pp. 818-824.

3 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, "Model Transformation with a Visual Control Flow Language", International Journal of Computer Science (IJCS), Number 1, Volume 1, 2006, ISSN 1306-4428, pp. 45-53.

4 **G. Mezei**, L. Lengyel, T. Levendovszky, H. Charaf, "A Model Transformation for Automated Concrete Syntax Definitions of Metamodeled Visual Languages", Electronic Communications of the EASST, Volume 4, 2006, Graph and Model Transformation 2006

5 **G. Mezei**, T. Levendovszky, H. Charaf, "Formalizing the Evaluation of OCL Constraints", Acta Polytechnica Hungarica, Volume 4, Issue 1, 2007, ISSN 1785-8860, pp. 89-110.

6 **G. Mezei**, "Hatékony szakterület-modellezési technikák a távközlésben", Magyar Távközlés, Budapest, 2006, Accepted

7 **G. Mezei**, T. Levendovszky, H. Charaf, "Optimization Algorithms for OCL Constraint Evaluation in Visual Models", Periodica Polytechnica, 2007, Accepted

8 **G. Mezei**, T. Levendovszky, H. Charaf, "Restrictions For OCL Constraint Optimization Algorithms", Electronic Communications of the EASST, Volume 5, 2006, OCL for (Meta-)Models in Multiple Application Domains 2006

Publications in International Conference Proceedings

9 T. Levendovszky, L. Lengyel, **G. Mezei**, H. Charaf, "A Systematic Approach to Metamodeling Environments and Model Transformation Systems in VMTS", International Workshop on Graph-Based Tools (GraBaTs) Electronic Notes in Theoretical Computer Science, 2004.

10 **G. Mezei**, T. Levendovszky, L. Lengyel, H. Charaf, "A Flexible Attribute Instantiation Technique for Visual Languages", IASTED on SE, February 15-17, 2005, Innsbruck, Austria, pp. 355-359.

11 **G. Mezei**, T. Levendovszky, L. Lengyel, H. Charaf, "Multilevel Metamodeling - A Case Study", MicroCAD, March 10-11, 2005, Miskolc, pp. 321-326.

12 L. Lengyel, **G. Mezei**, T. Levendovszky, H. Charaf, "A Metamodel-Based Technique for Attribute Transformation in Visual Model Processors", 5th Internal Conference of PhD Students, Engineering Sciences II, Miskolc, Hungary, 14-20 August 2005, pp. 133-138.

13 L. Lengyel, T. Levendovszky, **G. Mezei**, B. Forstner, H. Charaf, "Metamodel-Based Model Transformation with Aspect-Oriented Constraints", International Workshop on Graph and Model Transformation (GraMoT), Electronic Notes in Theoretical Computer Science, Volume 152, Tallinn, Estonia, September 28, 2005, pp. 111-123.

14 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, "Control Flow Support in Metamodel-Based Model Transformation Frameworks", EUROCON 2005 International Conference on "Computer as a tool", Proceedings of the IEEE, Belgrade, Serbia & Montenegro, November 21-24, 2005, pp. 595-598.

15 L. Lengyel, T. Levendovszky, **G. Mezei**, B. Forstner, H. Charaf, "Towards a Model-Based Unification of Mobile Platforms", 4th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA-06), Dubai, Sharjah, UEA, March 8-11, 2006.

16 **G. Mezei**, T. Levendovszky, H. Charaf, "Traversing Processor for Metamodel Environments", 5th Internal Conference of PhD Students, August, 2005, Miskolc, Hungary, pp. 127-133.

17 **G. Mezei**, T. Levendovszky, H. Charaf, "A Presentation Framework for Metamodeling Environments", 4th Workshop in Software Model Engineering, 2005 October, Montego Bay, Jamaica.

18 **G. Mezei**, T. Levendovszky, H. Charaf, "Presentation Framework - an Environment for Editing Metamodels", 6th International Symposium of Hungarian Researchers, November, 2005, Budapest, Hungary, pp. 551-562.

19 **G. Mezei**, T. Levendovszky, H. Charaf, "A Flexible Graphics Framework for Modeling Environments", International Carpathian Control Conference, Miskolc, Hungary, 2005, pp. 383-389.

20 **G. Mezei**, T. Levendovszky, H. Charaf, "Visual Presentation Solution for Domain Specific Languages",, IASTED on SE, February, 2006, Innsbruck, Austria, pp. 194-199.

21 **G. Mezei**, T. Levendovszky, H. Charaf, "Implementing an OCL 2.0 Compiler for Metamodeling Environments", 4th Slovakian - Hungarian Joint Symposium on Applied Machine Intelligence, SAMI 2006, January, 2006, Herlany, Slovakia, pp. 544-555.

22 L. Lengyel, T. Levendovszky, T. Vajk, **G. Mezei**, H. Charaf, "Practical Uses of Validated Model Transformation", IEEE EUROCON 2007, Sept, 2007, Warsaw, Poland, accepted

23 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, "A Visual Control Flow Language for Model Transformation Systems", IASTED International Conference on SOFTWARE ENGINEERING, SE 2006, Innsbruck, Austria, February 14-16, 2006, pp. 194-199.

24 B. Forstner, L. Lengyel, T. Levendovszky, **G. Mezei**, I. Kelényi, H. Charaf, "Model-Based System Development for Embedded Mobile Platforms", 13th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS), Potsdam, Germany, March 27 - 30, 2006.

25 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, "A Visual Control Flow Language", 5th WSEAS Int.Conf. on Software Engineering, Parallel and Distributed Systems, SEPADS '06, Madrid, Spain, February 15-17, 2006, pp. 30-35.

26 **G. Mezei**, L. Lengyel, T. Levendovszky, H. Charaf, "Optimization Algorithms For OCL Compilers", 5th WSEAS Int.Conf. on Software Engineering, Parallel and Distributed Systems, SEPADS '06, Madrid, Spain, February 15-17, 2006, pp. 55-60.

27 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, "Execution Properties of a Visual Control Flow Language", 4th Slovakian - Hungarian Joint Symposium on Applied Machine Intelligence', SAMI 2006, January 20 - 21, 2006, Herlany, Slovakia, pp. 368-379.

28 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, "Control Flow Support in Model Transformation Frameworks: An Overview", MicroCAD, March 16-17, 2006, Miskolc, pp. 193-198.

29 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, "Weaving Constraint Aspects into Metamodel-Based Model Transformation Steps", MicroCAD, March 16-17, 2006, Miskolc, pp. 199-204.

30 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, "Model-Based Development with Strictly Controlled Model Transformation", The 2nd International Workshop on Model-Driven Enterprise Information Systems, MDEIS 2006, May 23-24, 2006, Paphos, Cyprus, pp. 39-48.

31 **G. Mezei**, T. Levendovszky, H. Charaf, "Customized Error Handling Suport for OCL Compilers", MicroCAD, March, 2006, Miskolc, pp. 211-217.

32 **G. Mezei**, T. Levendovszky, H. Charaf, "A Domain-Specific Language for Visualizing Modeling Languages", ISIM, April, 2006, Prerov, Czech Republic.

33 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, "Constraint Validation-Driven Visual Model Transformation", Automation and Applied Computer Science Workshop (AACS), June, 2006, Budapest, Hungary, pp 29-41.

34 **G. Mezei**, L. Lengyel, T. Levendovszky, H. Charaf, "A Model Transformation Method To Support Automated Concrete Syntax Definitions", Automation and Applied Computer Science Workshop (AACS), June, 2006, Budapest, Hungary, pp 41-53.

35 **G. Mezei**, T. Levendovszky, H. Charaf, "An optimizing OCL Compiler for Metamodeling and Model Transformation Environments, IFIP Working Conference on Software Engineering Techniques, October, 2006, Warsaw, Poland, pp 61-73.

36 **G. Mezei**, T. Levendovszky, H. Charaf, "Extending an OCL Compiler for Metamodeling and Model Transformation Systems: Unifying the Twofold Functionality", 10th International Conference on Intelligent Engineering Systems 2006, June, 2006, London, United Kingdom.

37 **G. Mezei**, T. Levendovszky, H. Charaf, "Automatized Concrete Syntax Definitions For Domain Specific Languages", The 7th International Conference on Technical, June, 2006, Timisoara, Romania, pp 47-52.

38 **G. Mezei**, T. Levendovszky, H. Charaf, "A Distribution Technique For Graph Rewriting and Model Transformation Sytems", IASTED on PDCN, February 15-17, 2007, Innsbruck, Austria

39 **G. Mezei**, T. Levendovszky, H. Charaf, "A New Formalism Technique For OCL", 5th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence and Informatics, Jan, 2007, Poprad, Slovakia, pp 117-128.

40 **G. Mezei**, T. Levendovszky, H. Charaf, "Integrating Model Transformation Systems and Asynchronous Cluster Tools", 7th International Symposium of Hungarian Researchers on Computational Intelligence, Nov, 2006, Budapest, Hungary, pp 307-318.

41 T. Mészáros, **G. Mezei**, T. Levendovszky, "Using Patterns in Domain-Specific Languages", 7th International Symposium of Hungarian Researchers on Computational Intelligence, Nov, 2006, Budapest, Hungary, pp 167-178.

42 **G. Mezei**, "Supporting Transformation-Level Parallelism in Model Transformation", Automation and Applied Computer Science Workshop (AACS), June, 2007, Budapest, Hungary.

43 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, T. Mészáros, Z. Benedek, "The Introduction of the VMTS Mobile Toolkit", Kassel, Germany, 2007, Accepted.

44 **G. Mezei**, T. Levendovszky, H. Charaf, "Attribute Algebra for N-layer Metamodeling", WSEAS Conference on WSEAS Int.Conf. on Applied Informatics and Communication, Athens, Greece, 2007.

45 **G. Mezei**, T. Levendovszky, H. Charaf, "OCLASM: Towards an ASM-based formalism for OCL", Extended abstract, The 14th International ASM Workshop, Grimstad, Norway, 2007.

46 T. Mészáros, **G. Mezei**, T. Levendovszky, "Domain Specific Modeling Environment for Modeling Mobile Communication", Internal Conference of PhD Students, Miskolc, Hungary, 2007.

47 **G. Mezei**, T. Levendovszky, H. Charaf, "Towards Parallel Model Transformations", Workshop on Multi-Paradigm Modeling, Nashville, Tennessee (USA), 2007, Accepted.

Technical Report

48 L. Lengyel, T. Levendovszky, **G. Mezei**, "Metamodell-alapú modelltranszformáció", Budapesti Műszaki és Gazdaságtudományi Egyetem, Automatizálási és Alkalmazott Informatikai Tanszék, Budapest, Hungary, 2005.

49 **G. Mezei**, T. Levendovszky, H. Charaf, "Restrictions for OCL constraint optimization algorithms", Edited by Technische Universität Dresden, MoDELS Conference, Genova, Italy, 2006.

50 **G. Mezei**, T. Levendovszky, H. Charaf, "Optimizing Refinement for Metamodeling Environments with OCL Constrains", Labroatoire d'Informatique de Robotique et de Microelectronique de Montpellier, MoDELS doctoral symposium, Genova, Italy, 2006.

6 Citations

[1] is cited by:

A. Balogh, D. Varró, "Advanced Model Transformation Language Constructs in the VIATRA2 Framework", ACM Symposium on Applied Computing, 2006, pp. 1280-1287.

[7] is cited by:

E. Biermann, K. Ehrig, C. Köhler, G. Kuhns, G. Taentzer, E. Weiss, "Graphical Definition of In-Place Transformations in the Eclipse Modeling Framework", Model Driven Engineering Languages and Systems, 9th International Conference, MoDELS 2006, Genova, Italy, October 1-6, 2006, LNCS Volume 4199, pp. 425-439.

[13] is cited by:

Anne Etien, Cedric Dumoulin, Emmanuel Renaux, "Towards a Unified Notation to Represent Model Transformation", Institut National de Recherce en Informatique et an Automatique, Rapport de recherche, 2007.

References

[ASM] Egon Böerger, Robert Stärk, "Abstract State Machines", Springer-Verlag, Germany, 2003

[Atho et al., 1998] Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, "Compilers Principles, Techniques, and Tools", Addison - Wesley,1988

[Atkinson and Kuhne, 2001] Colin Atkinson and Thomas Kühne, "The Essence of Multilevel Metamodeling", UML 2001 - The Unified Modeling Language. Modeling Languages, Concepts, and Tools, 2001, pp. 19-33

[DSML] Domain-Specific Modeling, http://www.dsmforum.org/

[Ehrig et al., 2005] Hartmut Ehrig, Karsten Ehrig, Ulrike Prange, and Gabriele Taentzer, "Fundamentals of Algebraic Graph Transformation", Springer Verlag, 2005.

[Kumar et al., 1994] V. Kumar and A. Y. Grama and Nageshwara Rao Vempaty, ´´Scalable Load Balancing Techniques for Parallel Computers", Journal of Parallel and Distributed Computing, Volume 22, 1994, pp. 60–79

[Lengyel, 2006] László Lengyel, "Online Validation of Visual Model Transformations", PhD Thesis, Budapest, Hungary, 2006

[Levendovszky, 2005] Tihamér Levendovszky, "Applying Metamodels in Software Model Transformation Methods", PhD Thesis, Budapest, Hungary, 2006

[Levendovszky, 2006] Tihamér Levendovszky, Ulrike Prange, Hermut Ehrig, "A termination criteria for dpo transformations with injective matches", In Graph Transformation for Verification and Concurrency, Bonn, Germany, 2006

[MIC] Jonathan Sprinkle, "Model-Integrated Computing", IEEE Potentials, Issue 23(1), 2004.

[MOF] Meta Object Facility Specification, http://www.omg.org/mof/

[OCL] Jos Warmer, Anneke Kleppe, "Object Constraint Language, The: Getting Your Models Ready for MDA", Addison Wesley, 2003

[OCLFormalism] Object Management Group Object Constraint Language Specification, http://www.omg.org/docs/ptc/03-10-14.pdf

[Pierce, 1991] Benjamin C. Pierce, "Basic Category Theory for Computer Scientists", MIT Press, 1991.

[Rozenberg, 1997] Grzegorz Rozenberg, "Handbook on Graph Grammars and Computing by Graph Transformation: Foundations", World Scientific, Singapore, 1997.

[Soley et al., 2004] Richard Soley and David S. Frankel and John Parodi, "The MDA Journal: Model Driven Architecture Straight From The Masters", Meghan Kiffer Pr, 2004

[Sztipanovits, 1997] János Sztipanovits, Gábor Karsai, "Model-Integrated Computing", IEEE Computer, 1997, Volume 30, Issue 4

[Stark et al., 2001] Robert Stärk and Joachim Schmid and Egon Börger, "Java and the Java Virtual Machine: Definition, Verification, Validation", Springer Verlag, 2001

[UML] Object Management Group Unified Modeling Language 2.0 Specification, http://www.uml.org/#UML2.0

[VMTS] VMTS Homepage, http://www.vmts.aut.bme.hu