Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics

Applying Metamodels in Software Model Transformation Methods

Main Results of the Ph.D. Thesis

Tihamér Levendovszky

Advisor:
Dr. Hassan Charaf Ph.D.
Associate Professor

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

Budapest, 2005

Main Results of the Ph.D. Thesis

Tihamér Levendovszky

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

Goldmann György tér 3.
Budapest,
H-1111

E-mail: tihamer.levendovszky@aut.bme.hu
Phone: 463-1662
Fax: 463-3478

Advisor:

Dr. Hassan Charaf Ph.D.
Associate Professor

1. Preliminaries and Objectives

An increasing tendency of the linguistic abstraction level can be observed in the field of software engineering. The assembly languages that dominated the early phases were replaced by structured, and later on by object oriented programming languages. These paradigms help software development with several high-level language constructs, and with decreasing the semantic gap between the tasks specified in a natural language and the implementation offered by the programming languages. The explanation for this phenomenon is that the requirements are almost unlimited both in complexity, development time and the variety of the supported platforms.

The rise of the abstraction can be observed in case of the **visual modeling languages**, which appeared first as a phenomenon associated with the object oriented paradigm, and now they form an individual programming language in the OMG standard **Model-Driven Architecture** (MDA) [OMG MDA] [OMG MDAGuide] [Kleppe et al. 2003] [Mellor et al. 2004]. According to the vision of MDA, the developers build and specify precisely a Platform Independent Model (PIM), mainly in a graphical modeling language called Unified Modeling Language (UML) [OMG UML 2003] [Pender & Pender 2003]. Providing the platform specific information, the developers use model compilers to create a Platform Specific Model (PSM) automatically. PSMs can be executed on the given platform. Therefore, MDA needs efficient model compilers that can create code or other models from graph-based models.

Most of the model compilers use graph theoretical algorithms to traverse the model that must be processed (Traversing Model Processors, TMP), therefore, they interpret the model (Model Interpreters, [Lédeczi et al. 2001]). Meanwhile developing such model compilers are an error-prone, long work. Moreover, raising the abstraction level to the graphical model specification (Visual Model Processors, VMP) seems quite natural in a graphical, model-oriented development. A method called graph rewriting [Ehrig 1979] [Blostein et al. 1995] [Ehrig & Taenzer 1996] [Rozenberg 1997] [Ehrig et al. 1999] [Baresi & Heckel 2002]) provides an excellent background to specify VMPs. Graph rewriting consists of rules, which are quite similar to the production rules used in string grammars. As far as graph rewriting is concerned, a rule has a left-hand side (LHS). An isomorphic subgraph to the LHS graph must be found in the graph to be transformed. Then, the found subgraph must be replaced with the right-hand side of the rule. Graph rewriting has many branches, in case of software models the algebraic approaches [Ehrig 1979] [Rozenberg 1997], namely, the double pushout (DPO) and the single pushout (SPO) approaches turned out to be the most useful. These methods use abstract algebraic and category theoretical [Pierce 1991] [Barr & Wells 1999] constructs to formalize the process of rewriting. Considering the DPO approach, the identification condition must be enforced, which states that, on deletion, different vertices in the production rule cannot match the same vertex in the host graph. Furthermore, the dangling edge condition must be dealt with as well: if a vertex should be deleted which is connected to an edge that is not inside the match, the production rule cannot be fired. The SPO approach does not have such conditions, which would lead to contradictory operations, but the SPO approach resolves the situation by giving priority to the deletion.

Most of the tools that support graphical languages have been written for a specific graphical language family. In this case the specific program code makes the extension as well as the support for new languages nearly impossible. There is, however, a demand for modeling environments to support new domain specific languages (DSL), and to process them, especially, in the development of embedded systems. An efficient solution is to apply **metamodeling technique**s, which defines the constraints in a graphical model (metamodel) that must be satisfied along with the elements that may be contained in the models. It is possible to draw a parallel between metamodels and UML class diagrams: one creates an UML class diagram in the modeling environment (metamodel), and the environment allows creating only the object diagrams (model) that instantiate the lass diagram in modeling mode. Extending this technique with graphical features, we can easily and quickly create an environment that can support new modeling languages. The applications of this technique is not limited to the domain specific languages, since the gap between the versions of the UML standard is quite large, and the up-to-date support of the standard modeling language can be ensured by metamodeling techniques. Out of the metamodeling environments, the results of the dissertation were affected the most by the Generic Modeling Environment (GME) [Lédeczi et al. 2001] developed at Vanderbilt University, since I had the opportunity to take part in the development of this tool for a short period of time.

Therefore, it is worth creating visual model compilers to accelerate the development, which have formalism close to the standard and well-known UML, incorporate the advantages of graph rewriting and metamodeling. Taking these aspects into account, my goals were the following:

1. Create a metamodeling environment which is based on the experiences obtained from GME, but with direct support for developing VMP and with more efficient UML support (GME has primarily been developed to support DSLs).
2. Develop a graphical modeling language to describe the transformation rules, which is similar to UML, and takes it into account that the metamodels are available before the transformation as along with the general constraints in the metamodel.
3. Create an algorithm that executes the rewriting
4. Validate the rules based on the available metamodels.
5. Extend the existing mathematical results to the metamodel-based rewriting.
6. Investigate the possibility of the parallel execution of the rules based on the mathematical results.

Achieving these goals, one can simply create and process graphical modeling languages, moreover, the execution the rules can be performed in a parallel environment. Furthermore, the order of the rules can be optimized. This facilitates the development of mobile telephone software supported by automated generators, and transforming statecharts of embedded systems into executable code, and many other applications.

2. Methodological Summary

Since it is fundamental in software engineering that the theoretical results can be applied, and the problems created by the feedback from practice are addressed. Thus, we developed the **Visual Modeling and Transformation System** (VMTS) tool at the Department of Automation and Applied Informatics at Budapest University of Technology and Economics, which provided a background to examine the **empirical aspects**. This approach was characteristic to the whole research. The VMTS can be regarded as a **practical result**. I received similar kind of help from the GME [Lédeczi et al. 2001] and GReAT [Karsai et al. 2003] systems to form the theoretical results and raise open issues.

The methods of the research dealing with the theoretical results were primarily determined by the mathematical formalism available for the given problem, along with the related methods.

If the elements of the UML class diagram (metamodel) appear on LHS and RHS of the rules in the DPO approach, we must find an instantiation of LHS in the graph that must be transformed, and the result of the rewriting step is an instantiation of RHS. Since the task of subgraph isomorphism has been replaced by that of the instantiation, a novel algorithm must be developed. I have chosen the VF2 graph isomorphism algorithm [Cordella et al. 1999] [Cordella et al. 2001] [Foggia et al. 2001] to solve this problem, and I have extended it to find the instantiation. Moreover, I have introduced optimization steps in the execution of the algorithm. Moreover, I have investigated the fundamental properties of the instantiation as described in the UML standard [OMG UML], including the allowed multiplicity combinations on the instance level. Here I used the formalism of **graph theory** (set theory). Both the models and metamodels can be modeled directed, labeled graphs, where the label contains the type information, the instantiation relationships, the attributes and the presentation information. The latter two are not significant from the theoretical point of view. Considering the allowed instantiations, I used **linear algebraic formalism**, since the graph representation was proven to be difficult to deal with. I applied the incidence matrix representation of the graphs, and I have extended it to handle the multiplicity values, hence the elimination algorithm in Thesis II uses linear algebraic formalism. Since the multiplicities are positive integers, I have also used simple **number theoretical** formulae.

The availability of the metamodels means additional information, thus, it is worth examining how we can use this fact in validating the topology of the rules. The existing formal models can help in the solution, provided that they are extended to the metamodel-based case. Thesis III connects the existing theoretical models and the metamodels, moreover it provides conditions for the topological correctness of the rules. This is done mainly on a graph theoretical basis, whereas the homomorphism and graph homomorphism is an **abstract algebraic** contribution. This is necessary, because the proofs included in Thesis IV is built on category theoretical formalism, which does not examine the internal structure of the objects, but expects the satisfaction of several conditions, which can be proved by graph theoretical and abstract algebraic results.

Thesis IV is different from the other theses in a sense that the applied formalism is provided by **category theory**. The reason is that the available mathematical background, the DPO approach, is built on category theory. However **abstract algebraic result**s are also used in the proofs.

3. Novel Scientific Results

The results delineated in this section have been motivated by two main objectives. First, I wished to achieve results that are ensured by strict mathematical formalism; second, the results should be applicable in practice. The former had been achieved by formal models, the latter had been ensured by a more expressive visual transformation procedure which is applied in a software package. The novel results are divided into four theses.

Thesis I contains the practical results provided by VMTS. I have developed a pattern matching algorithm as a result of Thesis II. The algorithm is simplified by the knowledge about the number of objects participating in the valid instantiations. I have created the Incidence Matrix with Multiplicity (IMM) algorithm, and I have proven its correctness. I have also proven the properties of the instantiation used in the algorithm within Thesis II. Thesis III transforms the relationship between the instance and the metagraph into homomorphism. I have given two conditions for the type preserving homomorphic mapping in the first two subtheses. I have provided an algorithm which creates a new metagraph for an arbitrary metagraph, where the inverse relationship of the instantiation is a homomorphic mapping between an arbitrary instance graph and the resulting metamodel. Based on an equivalence definition that is applicable in practice, I have shown that the resulting metagraph is equivalent to the original metagraph. The homomorphic mapping cannot be established directly, thus, I have given a decomposition algorithm, where the instantiation relationship between the resulting graphs and the metagraph is homomorphism. Thesis IV generalizes the DPO approach: it proves that the metasquares in the categorical diagram form double pushouts under specific conditions, and do not violate the dangling edge condition. Finally, I prove a generalization of the parallelism theorem for metamodel-based rules.

1 Thesis I

I have designed a framework (Visual Modeling and Transformation System, VMTS) with a modular architecture. Furthermore, I have shown the following propositions via the VMTS software package:

- The class diagram, use case diagram, and statechart diagram can be realized by n-layer metamodeling techniques, using only one instantiation relationship, namely, the one between MOF M0 and M1 layers.

- Feature models and resource editors can also be realized with these techniques.
- The UML class diagram can be applied as a description language of the left-hand side and the right-hand side of the transformation rules, which has provided a solution to several practical issues.
- The description of the transformation and the creation of the transformation can benefit from the tight integration of graph transformation systems and metamodeling environments.
- I have illustrated the theoretical results with the help of VMTS.

Related publications:
[2][4][8][16][17][[19][20][21][22][23][24][25][28][29][30].

The VMTS software package has been used not only for illustrations, but it is the basis of several R&D project at DAAI. VMTS has been provided with several plug-ins in these projects, which aims at the following objectives:

- Supporting generative techniques
- Code generation form UML class and statechart diagrams
- Support for the development of mobile applications

2 Thesis II

I have created an algorithm, which determines the number of objects in the valid instantiations, and I have proven the correctness of the algorithm. I have given a pattern matching algorithm for the left-hand-side of the metamodel-based rules, along with an algorithm that provides conditions to split the search space of the pattern matching algorithm. Furthermore, I have extended the VF2 algorithm to metamodel-based pattern matching, along with heuristics.

Related publications: [5][7][9][12][13][14][15][19][21].

2.1 Definition (Labeled directed graphs, LDG):
Let Ω_V and Ω_E be two given alphabet for node and edge labels, respectively. Then the labeled directed graph is a six-tuple:

$$G = \left\langle G_V, G_E, s^G, t^G, lv^G, le^G \right\rangle, \tag{1}$$

where G_V is the set of vertices, G_E is the set of edges; s^G and t^G are the source and target functions ($s^G, t^G : G_E \to G_V$), which map an edge to its source and target vertices, respectively; and finally lv^G : $G_V \to \Omega_V$ and le^G : $G_E \to \Omega_E$, which assign a label to a vertex and an edge from the appropriate alphabet.

2.1 Subthesis II.1

I have proven that the class diagram depicted in Figure 1 can be instantiated by *na'* objects of type A and *nb'* objects of type B, where *n* is an arbitrary positive integer; *a* and *b* is the multiplicity values shown in Figure 1. GCD denotes the greatest common divisor.

$$\begin{aligned} a' &= \frac{a}{\mathrm{GCD}(a,b)} \\ b' &= \frac{b}{\mathrm{GCD}(a,b)} \end{aligned} \tag{2}$$

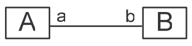
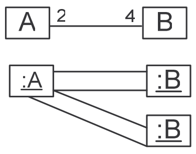


Figure 1. An association with nonzero multiplicity values

Subthesis II.1 also regards parallel edges. This is shown in Figure 2.



2. ábra. Párhuzamos linkek

If we do not allow parallel edges, the multiplicity constraints become stricter. This is addressed by subthesis II.2.

2.2 Subthesis II.2

I have proven that if no multiple edges are allowed for associations in the object diagram, the class diagram in Figure 1 can be instantiated by na objects of type A and nb objects of type B, where n is an arbitrary positive integer.

It is a negative result that, in general, only a disadvantageous upper bound can be given, since there exists a construct, where all the instances of a class on either side must be considered. The dissertation proves with a specific construct that there exists a model-metamodel pair, which reaches this upper bound.

2.3 Subthesis II.3

I have shown that the worst case number of objects that must be examined in order to decide the valid instantiation is in general

$$\#\mathrm{A}+\#\mathrm{B}, \tag{3}$$

and the greatest value of the corresponding number of objects:

$$\mathrm{n}=\min\left[\frac{\#\mathrm{A}}{\mathrm{a}},\frac{\#\mathrm{B}}{\mathrm{b}}\right], \tag{4}$$

where $\#\mathrm{A}$ denotes the number of *A* objects, and $\#\mathrm{B}$ denotes the number of *B* type objects.

After examining the case with one association, I generalized the results to the general case containing an arbitrary number of associations. I extended the incidence matrix to handle multiplicities (Incidence Matrix with Multiplicities, IMM). I have specified this formalism with its creator algorithm (Algorithm 2.1).

2.1 Algorithm CREATEIMM( )

```
1 Traverse the model graph
2 Take each edge $e_j$.
3 If vertices $v_k$ and $v_l$ are the incident upon $e_j$, then set $IMM_{kj}$ to the $v_k$ side multiplicity and $IMM_{lj}$ to the $v_l$ side multiplicity of $e_j$.
```

Changing to the linear algebraic representation, I have created an elimination algorithm, which terminates with error, if the metamodel represented by a given IMM cannot be instantiated, or provides a positive integer for each variable, whose integer multiples can participate in a valid instantiation (Algorithm 2.2).

2.2 Algorithm ELIMINATION(IMM IMM)

```
ELIMINATION(IMM imm)
 1 for each j column index
 2   r0=index of row with first nonzero element
 3   r1=index of row with second nonzero element
 4   if there is no r0 and r1 then break
 5   lcm=LCM(imm[r0,j],imm[r1,j])
 6   MultiplyRow(r0, lcm/imm[r0,j])
 7   MultiplyRow(r1, lcm/imm[r1,j])
 8   for each j2 column index
 9     if imm[r0,j2]= = imm[r1,j2] then
10        imm[r1,j2]=0
11     else if imm[r0,j2]!=0 and imm[r1,j2]!=0 then
12       Inconsistent parallel paths. No instantiation.
13     else if imm[r1,j2]!=0
14        imm[r0,j2]=imm[r1,j2];
15        imm[r1,j2]=0;
16     end if
17   end for
18 end for
19 rowlcm=LCM of the 0th row of imm
20 for each j column index
21   imm[0,j]=rowlcm/ imm[0,j]
22 end for
```

We have implemented the algorithm in VMTS, and its source code can be downloaded from [VMTS]. I have shown that the IMM algorithm retrieves all the possible numbers of objects participating in a valid instantiations, and the algorithm is correct.

2.4 Subthesis II.4

In the IMM algorithm, the matrix resulted through an arbitrary elimination step represents an equation set which is equivalent to the multiplicity equations established according to the instantiation equation. Furthermore, there is no solution which can form a valid instantiation, but it is not among the results of the IMM algorithm.

Investigating the existing algorithms, I have chosen the VF2 algorithm as a basis of my pattern matching algorithm. Since the metamodel is available as additional information, I have extended the VF2 algorithm to match the type information, and I applied heuristics efficient in most cases, but never worse than VF2. These heuristics can be required explicitly on the VMTS user interface, thus, the designer can directly ensure the efficiency.

From now on, G^H denotes the graph that is a subject of pattern matching, G^{LHS} is the pattern graph. The edge set of the graph is denoted by E, the node set is represented by V, the upper index refers to the appropriate graph. Variables $n \in G^H, m \in G^{LHS}$ are two vertices to match, $M \subseteq V^H \times V^{LHS}$ is a mapping between the pattern and a subgraph of G^H. The upper index denotes the appropriate side. The multiplicity of the element *x*,*y* in a multiset is denoted by $\mu^{MS}(x, y)$. Sets $pred(G, x)$ and $succ(G, x)$ denotes the neighbors reachable via directed edges from the vertex *x*, respectively: *pred* means the neighbors having the incoming edges to vertex *x*, whereas *succ* means the neighbors reachable via outgoing edges from *x*. The functions *typeof* retrieves the type of an element. Sets *T* mean the set of the nodes adjacent to the actual *m* and *n* nodes, where the *out* index denotes connectivity by the outgoing edges, *in* represents the connectivity by the incoming edges. The lower index refers to the graph containing the nodes or the edges.

I have defined the type matching for the edges as follows:

$$\begin{aligned} &m' \in M^{LHS} \wedge \exists n' | (n', m') \in M, n' \in pred(G^{H}, n) \wedge \\ &\forall (typeof(n, n')\mu^{E^{LHS}}(m', m) \leq \mu^{E^{H}}(n', n) \wedge typeof(m', m) = typeof(n' n) \end{aligned} \tag{5}$$

$$\begin{aligned} &m' \in M^{LHS} \wedge \exists n' | (n', m') \in M, n' \in succ(G^{H}, n) \wedge \\ &\forall (typeof(n, n')\mu^{E^{LHS}}(m, m') \leq \mu^{E^{H}}(n, n') \wedge typeof(m, m') = typeof(n, n') \end{aligned} \tag{6}$$

I have replaced the constraint checking part of the VF2 by the following procedure: (Algorithm 2.3):

2.3 Algorithm

```
SIMPLE_FEASIBLE (Mapping M, Node n, Node m)
1 if not typeof(n)=typeof(m) return false;
2 for each predecessor m’ of m
3     do if not Eq. (5)
4         return false
5     end if
6 end for
7 for each successor m’ of m
8     do if not Eq. (6)
9         return false
10    end if
11 end for
```

The novel algorithm is referred to as MetaVF2. I have modified the part of VF2 that finds the candidate parts for the matching as follows (Algorithm 2.4):

2.4 Algorithm

```
COMPUTE_CANDIDATE_PAIRS()
1 if $T_H^{out} \neq \emptyset \wedge T_{LHS}^{out} \neq \emptyset$ then $P = T_H^{out} \times \{\min T_{LHS}^{out}\}$
2 else if $T_H^{in} \neq \emptyset \wedge T_{LHS}^{in} \neq \emptyset$ then $P = T_H^{in} \times \{\min T_{LHS}^{in}\}$
3 else $\mathrm{P} = \mathrm{Instanceof}(\min(\mathrm{V}^{\mathrm{LHS}} - \mathrm{M}^{\mathrm{LHS}})) \times \{\min(\mathrm{V}^{\mathrm{LHS}} - \mathrm{M}^{\mathrm{LHS}})\}$
```

2.5 Subthesis II.5

I have proven that the worst case of the COMPUTE_CANDIDATE_PAIRS() procedure is the one provided by the VF2 algorithm, assuming connected pattern graph.

It seems natural that the algorithm chooses the metamodel element having the largest number of instances. This is not always optimal.

2.6 Subthesis II.6

I have shown that if we prefer to choose the V^{LHS} metaelement having the most instances in the next step of the algorithm, the termination of the algorithm can require more steps than the *MetaVF2* algorithm.

Applying the results detailed above, I have created an algorithm that performs the pattern matching with instantiation, which was the expected objective. This algorithm is the central algorithm of the VMP engine included in VMTS.

3 Thesis III.

I have supplied a method for the topological validation of the metamodel-based rules assuming that the metamodels of the input model and the output model are available. I have provided a necessary and sufficient condition for the existence of homomorphic instantiation. I have proven that for each metamodel, there exists an equivalent metamodel with a homomorphic *typeof* mapping. I have shown that an instance can be decomposed into non-isomorphic subgraphs with homomorphic instantiation mapping.
Related publications: [3][6][9][18][21][30].

I have supplied a theoretical background for the topological validation of the given transformation steps based on the metamodels of the input and output models. Furthermore, these results provide the background to generalize the existing category theoretical results (DPO approach) to metamodels. Since the arrows in category **Graph** are graph homomorphisms, the *typeof* mapping must be converted into homomorphisms.

3.1 Definition (Graph homomorphism for directed and labeled graphs):
A graph homomorphism $f : G \to H$ is a pair of functions: $f = \langle f_V : G_V \to H_V, f_E : G_E \to H_E \rangle$, which preserves the sources, targets, and labels, that is, it satisfies the following conditions:

- $f_V \circ t^G = t^H \circ f$
- $f_V \circ s^G = s^H \circ f_E$
- $lv^H \circ f_V = lv^G$
- $le^H \circ f_E = le^G$

3.2. Definition (Equivalence, compatibility, and partial compatibility of metamodels):
Let *Meta1* and *Meta2* be two LDGs, with labels conforming to the UML class diagram. *Meta1* and *Meta2* are **equivalent** if and only if any LDG provided with labels conforming to the UML object diagram i.e. any instance of *Meta1* is also an instance of *Meta2*, and vice versa. If only one direction is satisfied, namely, all instances of *Meta1* are the instance of *Meta2*, then *Meta2* is **compatible** with *Meta1*. Hence, equivalence can be defined as bidirectional compatibility. If at least one instance of *Meta1* - that instantiates each element in *Meta1* - is contained by at least one instance of *Meta2*, then *Meta2* is **partially compatible** with *Meta1*.

Considering the instantiation, I have proven the following subtheses:

3.1 Subthesis III.1

Let *Meta* be an LDG with labels conforming to the UML class diagram. Let another LDG *Instance* be given with labels conforming to the UML object diagram. Furthermore, we assume that *Instance* instantiates *Meta*, according to the instantiation rules enforced by UML. If *Meta* does not contain inheritance relationship (nor abstract classes, which would be semantically meaningless without inheritance), then the TPM between the *Instance* and *Meta* graph is a graph homomorphism.

3.2 Subthesis III.2
Let *Meta* be an LDG with labels conforming to the UML class diagram. Let another LDG *Instance* be given with labels conforming to the UML object diagram. In addition we assume that *Instance* instantiates *Meta* according to the instantiation rules enforced by UML. Let A_n denote the set of the associations connected to the *n*th layer of a class hierarchy. Let L_n represent the set of the links instantiating elements of A_n. *Instance* can be mapped to *Meta* via graph homomorphism if and only if no :O object is attached to link from more than one L_n set, where A_n belongs to the class hierarchy of *C*, which is the type of :O.

Consequently, the general UML class diagram cannot be instantiated by graph homomorphisms. Therefore, I constructed a metamodel that is as much expressive and applicable as the original, but the instantiation is a homomorphic mapping. Hence, I introduced an equivalence definition (Definition 3.2), which specifies the practical aspects of the equivalence. Then, I created an algorithm which creates a metamodel compatible with the original input metamodel, but the inverse of the instantiation is homomorphism. A sample input and output of the algorithm is depicted in Figure 3.

3.1 Algorithm (outline)
1. Walk through the inheritance hierarchy and copy all inherited data (attributes associations) from the ancestors to the derived class, while the value of the variable multiplicities are taken care of.
2. Remove inheritance.
3. Remove all abstract classes.

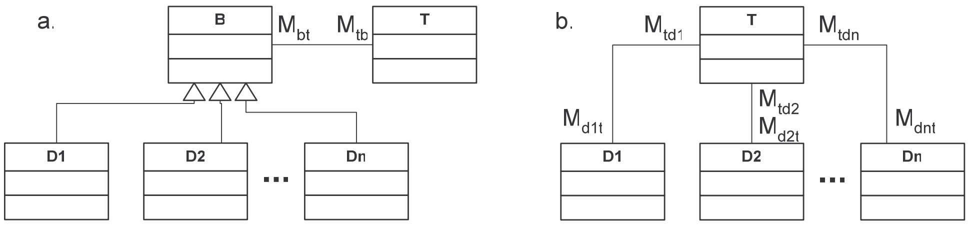


Figure 3. Generator metamodel (a) and the related homomorhic metamodel (b)

I have proven the following property of Algorithm 3.1.

3.3 Subthesis III.3
I have shown that the homomorphic metamodel and its generator metamodel (the input of Algorithm 3.1) are equivalent.

3.4 Subthesis III.4
Let *Meta1* and *Meta2* be an LDG, with labels conforming to the UML class diagram, both homomorphic metamodels. If *Meta1* is compatible with *Meta2*, *Meta1* is a (not always connected) subgraph of *Meta2* not regarding the multiplicity labels. The following formula is always true:

$$M_{Meta1} \subseteq M_{Meta2}, \qquad (7)$$

where $\mathrm{M}_{\mathrm{Meta1}}$ and $\mathrm{M}_{\mathrm{Meta2}}$ are the sets of the allowed multiplicity for the *Meta1* and *Meta2* side respectively at the same topological position. In case of partial compatibility, where the zero multiplicity values are not allowed, it is enough to enforce the following conditions for each corresponding multiplicity pairs:

$$\mathrm{SupM}_{\mathrm{Meta1}} \leq \mathrm{SupM}_{\mathrm{Meta2}}, \tag{8}$$

$$\mathrm{M}_{\mathrm{Meta1}} \cap \mathrm{M}_{\mathrm{Meta2}} \neq \varnothing. \tag{9}$$

where Sup is the supremum (least upper bound) of the set which contains the allowed multiplicity values.

3.5 Subthesis III.5

Each I instance graph can be decomposed into $I_1, I_2 ... I_K$ non-isomorphic subgraphs such that the instantiation relationship between the homomorphic M metagraph having nonzero multiplicities on the association ends; the $I_k, (k = 1..K)$ graph is a graph homomorphism, where I is an instance of M .

3.6 Subthesis III.6

I have proven that the worst case complexity of the decomposition algorithm related to Subthesis III.5 is $O(\#E(I)\#E(M))$ as a function of the number of edges in the instantiation and the metagraph.

The results included in Thesis III facilitate the topological validations of the rules, comparing them to the metamodels of the input and output models. It also contains results for correspondences between compatibility and subgraph isomorphism. I found the applications of the thesis the most efficient when the conditions presented in Subthesis III.2 are fulfilled, in this case fast pattern matching and constraint validating algorithms can be constructed.

4 Thesis IV

Based on the results of the DPO approach, I have proven that if the metamodels do not contain elements in common, the execution order of the metamodel-based rules is indifferent, and they can be executed in parallel, and if the execution order of two rules can be altered, then their parallel execution does not change the elements rewritten by the other rule. I have shown that the other direction of this statement is also true.

The category theoretical approach of this thesis is fundamentally different from the graph theoretical and abstract algebraic approaches of the two previous theses. Unfortunately, the existing methods (DPS, SPO) cannot apply pure categorical constructs, but the dominating approaches and the generalizations takes this direction, for example, the High-Level Replacement Systems. Therefore, I used abstract algebraic results in the proofs, accompanying the categorical constructs.

A main result of the DPO approach is the Parallelism Theorem. The parallel rule applications were most important in VMTS, thus, I have shown two fundamental results: (i) the DPO approach can be generalized to the metamodel-based cases, and then the metamodels do not violate the DPO gluing conditions, furthermore, (ii) how the Parallelism Theorem can be applied to the metamodel-based LHS.

Related publications: [3][6][9][21][30].

4.1 Defintion (*Graph* Category)

- Objects: LDGs,
- Arrows: LDG homomorphisms,

- Domain: the domain of the homomorphism; the codomain is the range of the homomorphism.
- Composite operator: composite of graph homomorphisms.
- The identity arrows are the identity functions for both vertices and edges.

4.2 Definition (graph production):
A graph production rule $p:(L \xleftarrow{l} K \xrightarrow{r} R)$ is composed of a production name *p*, and a pair of injective graph morphism: $l:K \to L$, és $r:K \to R$. The graphs L, K, and R are called the left-hand side, the interface graph[1] and the right hand side graph of *p*, respectively.

4.3 Definition (pushout, pushout complement):
Let C be a category, with two given arrows $b:A \to B, c:A \to C$. The $\langle D, g:B \to D, f:C \to D \rangle$ triple is called the pushout of $\langle b,c \rangle$ if the following conditions hold:

1. $g \circ b = f \circ c$ (commutative property)
2. For any object D' and arrows $g':B \to D'$, and $f':C \to D'$ if $g \circ b = f \circ c$, there exists a unique arrow $h:D \to D'$ such that $h \circ g = g'$ and $h \circ f = f'$ (universal property).

The triple (Figure 4) $\langle C, c:A \to C, f:C \to D \rangle$ is called the pushout complement of $\langle b, g \rangle$, and C is called $\langle b, g \rangle$ the pushout complement object of $\langle b, g \rangle$. D is called a pushout object of $\langle b, c \rangle$.

$$\begin{tikzcd}
A \arrow[r, "b"] \arrow[d, "c"'] & B \arrow[d, "g"] \arrow[dr, "g'"] & \\
C \arrow[r, "f"] \arrow[rr, bend right, "f"'] & D \arrow[r, dashed, "h"] & D'
\end{tikzcd}$$

Figure 4. Pushout

The statements below use the notations from Figure 6, where *G* denotes the input graph, *H* represents the output graph, and the *M* suffix refers to the metamodels.

4.4 Definition (Proper metamodels)
If an LM and an RM satisfy the condition of partial compatibility, elaborated in Proposition 5.4, with respect to GM and HM and they do not contain nonzero multiplicity values, they are called **proper metamodels**.

4.1 Subthesis IV.1
Let *O* be the common ancestor of each elements in the host graph metamodel. If the metamodel-based rewriting rule contains the subgraph depicted in Figure 5, then the node *X* and the related edges can be removed without violating the dangling edge condition

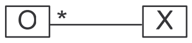


Figure 5. Subgraph for Subthesis IV.1

[1] K stands for the German *Klebegraph* ("gluing graph").

4.2 Subthesis IV.2

I have supplied a categorical model for metamodel-based transformation rules. Furthermore, I have shown that if the morphisms are inclusions in the outer double square, and the inner square is a double pushout, the outer double square is a double pushout if the following conditions hold:

$$V_{DM} = V_{GM} \setminus (V_{LM} \setminus V_{RM}) \tag{10}$$

$$E_{DM} = E_{GM} \setminus (E_{LM} \setminus E_{RM}) \tag{11}$$

$$V_{DM} = V_{HM} \setminus (V_{RM} \setminus V_{LM}) \tag{12}$$

$$E_{DM} = E_{HM} \setminus (E_{RM} \setminus E_{LM}) \tag{13}$$

Figure 6 uses the notations from Definition 4.2, and the inner square contains the appropriate subgraphs of the input graph (*G*), and the output graph (*H*). The *M* suffix denotes the metamodel of the given graph.

$$\begin{tikzcd}
LM \arrow[ddd, bend right=60, "lgm"'] & KM \arrow[l, "lkm"'] \arrow[r, "krm"] & RM \arrow[ddd, bend left=60, "rhm"] \\
L \arrow[u, "lm"] \arrow[d, "m"] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[u, "km"] \arrow[d, "d"] & R \arrow[u, "rm"] \arrow[d, "m^*"] \\
G \arrow[d, "gm"] & D \arrow[l, "l^*"'] \arrow[r, "r^*"] \arrow[d, "dm"] & H \arrow[d, "hm"] \\
GM & DM \arrow[l, "gdm"'] \arrow[r, "dhm"] & HM
\end{tikzcd}$$

Figure 6. Illustration for Subthesis IV.2

4.3 Subthesis IV.3

I have proven that if the L, K, R, G, D, H double square (Figure 6) is a double pushout, and the direct derivation does not violate the metamodel GM, then the diagrams in Figure 7 are pushouts.

$$\begin{tikzcd}
LM \arrow[d, "lgm"'] & LM \arrow[l, "lkm"'] \arrow[d, "lgm"] \\
GM & GM \arrow[l, "gdm"']
\end{tikzcd}
\qquad
\begin{tikzcd}
RM \arrow[r, "krm"] \arrow[d, "lhm"'] & RM \arrow[d, "lhm"] \\
GM \arrow[r, "dhm"] & GM
\end{tikzcd}$$

Figure 7. Illustration for Subthsis IV.3

4.4 Subthesis IV.4

If two LHS metamodels do not contain elements in common, then the order of the rules are invariant and can be executed parallel.

4.5 Subthesis IV.5

I have shown that if two rules given by metamodels can be applied sequential independently, the applications can be applied parallel independently as well. Moreover, if two rules given by metamodels are parallel independent, the applications are sequential independent as well.

Apart from the mathematical results of Thesis IV, it supplies conditions on a formal basis to execute the rewriting rules in parallel. Depending on the executing hardware, this can accelerate the transformation process significantly. From the mathematical point of view further existing results can be generalized to the metamodel-based case.

4. Application of the Novel Scientific Results

The novel scientific results appear in software packages, and in their applications. Since we can present the applicability to the industry based on the case studies mentioned below, the number of the applications are expected to increase with the growing interest. The related publications are as follows: [2][4][8][16][17][[19][20][21][22][23][24][25][28][29][30].

4.1. Software Packages

As it has already been mentioned, the applicability of the novel results are illustrated by a software package called VMTS. I appreciate the help of the following colleagues who contributed to the implementation of the system, namely, *László Lengyel*, *Gergely Mezei*, and *László Angyal*. Therefore, they have been my coauthors in several publications.

The success of the metamodel-based rewriting has been given by the initial versions of the GReAT transformation system, which we designed and implemented on the top of the UDM framework [Magyari et al. 2003] together with the researchers of ISIS at Vanderbilt University. This work inspired the [10][11][12][13] publications.

4.2. Applications

We have used the metamodel-based rewriting extended with multiplicity to solve several practical issues. Normalizing feature models [17] is a problem that arises in the field of generative programming [Czarnecki & Eisenecker 2000]. This transformation can be solved with minimal time and development effort with VMTS VMPs. Publications [8][19] present the procedures generate C++ code from statechart diagrams. These generators also use metamodel-based rewriting techniques extended with multiplicities. An example for code generation from UML class diagram to C# is elaborated in [23], whereas [19] generates code for Quantum Framework [Samek 2002] from UML statechart input. These applications are also implemented in VMTS.

We used the results of Thesis II to realize the transformation with the novel pattern matching algorithms. The applicability of the rules can be investigated by the results in Thesis III. Using the results included in Thesis IV, the system can recognize the possibility of parallel execution or optimize based on the invariant order of the rules, without the availability of the actual input model.

The *Simplian* research project conducted at the Department of Automation and Applied Informatics aims at creating rapid user interface development technologies for the *Symbian* operating system. The user interface is modeled by VMTS, the C++ code generation is performed by a VMP. Statecharts are also involved in UI modeling. The success of this sort of code generation has already been demonstrated by VMTS.

5 Related Publications

Books:

[1] Bányász G, **Levendovszky T**, Linux Programozás,
Szak Kiadó, 2003, ISBN 963 9131 57 1

Book Parts:

[2] Albert I, Balássy Gy, Charaf H, Erdélyi T, Horváth Á, **Levendovszky T**, Péteri Sz, Rajacsics T, A .NET Framework és programozása, Szak Kiadó, 2004, ISBN 963 9131 62 8

Journals:

[3] **Levendovszky T**, Lengyel L, Charaf H, Implementing a Metamodel-Based Model Transformation System, Buletinul Stiintific al Universitatii "Politehnica" din Timisoara, ROMANIA Seria AUTOMATICA si CALCULATOARE PERIODICA POLITECHNICA, Transactions on AUTOMATIC CONTROL and COMPUTER SCIENCE Vol.49 (63), 2004, ISSN 1224-600X , pp. 89-95

[4] **Levendovszky T**, Angyal L, Charaf H, Software Design Trade-Offs in Highly Configurable User Interface Construction, ROMANIA Seria AUTOMATICA si CALCULATOARE PERIODICA POLITECHNICA, Transactions on AUTOMATIC CONTROL and COMPUTER SCIENCE Vol.49 (63), 2004, ISSN 1224-600X. pp. 95-101

[5] **Levendovszky T**, Lengyel L, Charaf H, A UML Class Diagram-Based Pattern Language for Model Transformation Systems, WSEAS Transactions on Computers, Issue 2, Volue 4, February 2005, ISSN 1109-2750, pp. 190-195

[6] **Levendovszky T**, Lengyel L, Charaf H, Extending the DPO Approach for Topological Validation of Metamodel-Level Graph Rewriting Rules, WSEAS Transactions on Information Science and Applications, Issue 2, Volue 2, February 2005, ISSN 1790-0832, pp. 226-231

[7] **Levendovszky T**, Charaf H, Pattern Matching in Metamodel-Based Model Transformation Systems, Periodica Polytechnica Electrical Engineering, to be pulished

[8] Lengyel L, Levendovszky T, Charaf H, Constraint Validation Support in Visual Model Transformation Systems, ACTA CYBERNETICA, to be pulished

[9] Fill HG, Geiger L, Zündorf A, **Levendovszky T**, Lengyel L, Mezei G, Charaf H, A comparison of three tool-based approaches for the practical realization of UML statechart diagrams, Invited Paper, Software and System Modeling, Springer, submitted

[10] Lengyel L, **Levendovszky T**, Aspektus-orientált programozás, Híradástechnika, Volume LIX, 2004/10, pp. 8-12

[11] Lengyel L, **Levendovszky T**, Introduction to Aspect-Oriented Programming, Híradástechnika, Volume LX. 2005, pp. 18-23

Publications in International Conference Proceedings:

[12] **Levendovszky T**, Karsai G, Maroti M, Ledeczi A, Charaf H, Model reuse with metamodel-based transformations, Lecture Notes in Computer Science - ICSR7, Austin, TX, April 18, 2002, pp.166-178

[13] Agrawal A, **Levendovszky T**, Sprinkle J, Shi F, Karsai G, Generative Programming via Graph Transformations in the Model-Driven Architecture, OOPSLA - Workshop on Generative Techniques in the Context of Model Driven Architecture, Seattle, WA, November 5, 2002.

[14] Sprinkle J, Agrawal A, **Levendovszky T**, Shi F, Karsai G, Domain Evolution in Visual Languages Using Graph Transformations, OOPSLA - 2nd Workshop on Domain-Specific Languages, Seattle, WA, November 4, 2002.

[15] Sprinkle J, Agrawal A, **Levendovszky T**, Shi F, Karsai G, Domain Translation Using Graph Transformations, Tenth IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, Huntsville, AL, April 7, 2003., pp. 159-168

[16] Angyal L, **Levendovszky T**, Charaf H, Software Design with Generative Programming and Traditional Object-Oriented Techniques, microCAD, March 18-19, Miskolc, 2004, Miskolc. pp. 9-14

[17] Lengyel L, **Levendovszky T**, Charaf H, Supporting Round-Trip Engineering in Modeling Environments with the Application of Meta-Modeling Techniques, IASTED on SE, Innsbruck, 2004, pp. 178-182

[18] **Levendovszky T**, Charaf H, Parametrized Multiplicity Constraints in UML Like Metamodels, microCAD, March 18-19, Miskolc, 2004, Miskolc, pp. 287-292

[19] **Levendovszky T**, Lengyel L, Charaf H, Software Composition with a Multipurpose Modeling and Model Transformation Framework, IASTED on SE 2004, Innsbruck, 2004, pp. 590-594

[20] Lengyel L, **Levendovszky T**, Charaf H, Metamodel-Based Software Model Storage and Transformation System, microCAD, March 18-19, Miskolc, 2004, pp. 281-286

[21] **Levendovszky T**, Lengyel L, Mezei G, Charaf H, 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, to be pulished

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

[23] Lengyel L, **Levendovszky T**, Charaf H, Constraint Handling in Feature Models, 5th International Symposium of Hungarian Researchers on Computational Intelligence, Budapest, 2004, pp.279-290

[24] Lengyel L, **Levendovszky T**, Kozma P, Charaf H, Compiling and Validating OCL Constraints in Metamodeling Environments and Visual Model Compilers, IASTED on SE, February 15-17, 2005, Innsbruck, Austria, pp. 48-54

[25] Lengyel L, **Levendovszky T**, Charaf H, Weaving Crosscutting Constraints in Metamodel-Based Transformation Rules, 8th International Conference on Information Systems Implementation and Modeling, ISIM '05, April 19-20, 2005, Hradec nad Moravicí, Czech Republic, pp. 119-126

[26] Bogárdi-Mészöly Á, Imre G, **Levendovszky T**, Charaf H, Determining the Distribution of the Response Time in J2EE Web Applications, microCAD, March 10-11, Miskolc, 2005, pp.33-38

[27] Iváncsy R, **Levendovszky T**, Charaf H, .NET Facilities in Data Mining Applications, microCAD, March 10-11, Miskolc, 2005, pp. 167-172

[28] Lengyel L, **Levendovszky T**, Charaf H, Graph Transformation and Constraint Validation-Driven User Interface Handler Code, microCAD, March 10-11, Miskolc, 2005, pp. 267-272

[29] Lengyel L, **Levendovszky T**, Charaf H, Implementing a Metamodel-Based OCL-Compiler, microCAD, March 10-11, Miskolc, 2005, pp. 273-278

[30] **Levendovszky T**, Lengyel L, Charaf H, Creating a Homomorphic Instantiation Mapping Between Metamodels and the Model Parts, microCAD, March 10-11, Miskolc, 2005, pp. 279-283

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

[32] D. Bisztray, **Levendovszky T**, Charaf H, Defining Feature Modeling-Based Constraints, microCAD, March 10-11, Miskolc, 2005, pp. 19-24

[33] Lengyel L, **Levendovszky T**, Charaf H, Managing Crosscutting Constraints in Metamodel-Based Model Transformation Frameworks, International Carpathian Control Conference, ICCC'2005, Miskolc-Lillafüred, Hungary, May 24-27, 2005

[34] Lengyel L, **Levendovszky T**, Charaf H, Implementing an OCL Compiler for .NET, .Net Technologies 2005, Plzen, May 2005

[35] Lengyel L, **Levendovszky T**, Charaf H, Aspect-Oriented Constraints in Metamodel-Based Model Transformation, 5th Internal Conference of PhD Students, Miskolc, Hungary, 14-20 August 2005, pp. 109-114

[36] Lengyel L, **Levendovszky T**, Charaf H, A Case Study: Supporting Metamodel-Based Graph Transformation with Aspect-Oriented Constraints, 5th Internal Conference of PhD Students, Miskolc, Hungary, 14-20 August, 2005 pp.115-120

6. Citations

[12] is cited by:

Schloegel K, Oglesby D, Engstrom E, Towards Next Generation Metamodeling Tools, Second Workshop on Domain Specific Visual Languages, OOPSLA 2002

Tratt L, Clark T, Using Icon-derived technologies to drive model transformations, WiSME@UML'2003 - UML Workshop W2, San Francisco, USA, 2003

Tratt T, Clark T, Model transformations in Converge, Workshop in Software Model Engineering (WiSME) 2003

Appukuttan B, Clark T, Reddy S, Tratt L, Venkatesh R, A Pattern based model driven approach to model transformations, Metamodelling for MDA. University of York, UK. November 2003

Appukuttan B, Clark T, Reddy S, Tratt L, Venkatesh R, A model driven approach to building implementable model transformations,WiSME@UML'2003 - UML Workshop W2, San Francisco, USA, 2003

[13] is cited by:

Willink ED, A concrete UML-based graphical transformation syntax : The UML to RDBMS example in UMLX, Metamodelling for MDA. University of York, UK. November 2003

Boulet P, Dekeyser J-L, Dumoulin C, Marquet P, MDA for SoC Embedded Systems Design, Intensive Signal Processing Experiment" Model Driven Architecture in the Specification, Implementation and Validation of Object-oriented Embedded Systems, San Francisco, California, USA, 2003

Wagelaar D, Jonckers V, A Concept-Based Approach to Software Design. In proceedings of the 7th IASTED International Conference on Software Engineering and Applications (SEA 2003), Marina del Rey, USA, November 2003. ISBN/ISSN: 0-88986-394-6

Graw G, Herrmann P, Generation and Enactment of Controllers for Business Architectures Using MDA, Lecture Notes in Computer Science, Volume 3047, May 2004, Pages 148 – 166

Wagelaar D, Towards a Context-Driven Development Framework for Ambient Intelligence. In: Proceedings of the 24th International Conference on Distributed Computing Systems Workshops (ICDCS 2004 Workshops), IEEE Computer Society, 2004

Suvee D, Vanderperren W, Wagelaar D, There are no Aspects. Software Composition Workshop @ ETAPS 2004, Electronic Notes in Theoretical Computer Science, Elsevier Science, April 2004

[19] is cited by:

Cechticky V, Pasetti A, Rohlik O, Schaufelberger W, XML-Based Feature Modelling pp. 101-114, Software Reuse: Methods, Techniques and Tools: 8th International Conference, ICSR 2004, Madrid, Spain, July 5-9, 2009. Proceedings. Lecture Notes in Computer Science 3107 Springer 2004, ISBN 3-540-22335-5

7 References

[Baresi & Heckel 2002] Baresi L, Heckel R, Tutorial Introduction to Graph Transformation: A Software Engineering Perspective, In Corradini, A.,H. Ehrig, H.-J. Kreowski, G. Rozenberg (Eds): Proc. 1st Int. Conference on Graph Transformation (ICGT 02), Barcelona, Spain, Volume 2505 of Lecture Notes in Comp. Science. Springer-Verlag, October 2002.

[Barr & Wells 1999] Barr M, Wells C, Category Theory Lecture Notes for ESSLLI, www.folli.uva.nl/CD/1999/library/pdf/barrwells.pdf, 1999.

[Blostein et al. 1995] Blostein D, Fahmy H, Grbavec A, Practical Use of Graph Rewriting, Technical Report No. 95-373, Department of Computing and Information Science, Queen's University, Kingston, Ontario, Canada, January, 1995.

[Cordella et al. 1999] Cordella LP, Foggia P, Sansone C, Vento M, Performance Evaluation of the VF Graph Matching Algorithm, Proc. of the 10th ICIAP, IEEE Computer Society Press, vol. 2, pp. 1038-1041, 1999.

[Cordella et al. 2001] Cordella LP, Foggia P, Sansone C, Vento M, An Improved Algorithm for Matching Large Graphs, Proc. of the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition, Ischia, May 23-25, pp. 149-159, 2001.

[Czarnecki & Eisenecker 2000] Czarnecki K, Eisenecker UW, Generative programming: methods, tools, and applications, Addison-Wesley, 2000

[Ehrig & Taenzer 1996] Ehrig H, Taentzer G, Computing by Graph Transformation. A Survey and Annotated Bibliography Technical Report, TU Berlin, No. 96-21, 1996

[Ehrig 1979] Ehrig H, Introduction to the Algebraic Theory of Graph Grammars, In:Graph Grammars and Their Applications to Computer Science and Biology, Springer, Ed. Claus V, Ehrig H, Rozemberg G, Berlin, 1979.

[Ehrig et al. 1999] Ehrig H, Engels G, Kreowski H-J, Rozemberg (ed.), Handbook on Graph Grammars and Computing by Graph Transformation: Application, Languages and Tools, Vol.2. World Scientific, Singapore, 1999.

[Foggia et al. 2001] Foggia P, Sansone C, Vento M, A Performance Comparison of Five Algorithms for Graph Isomorphism, Proc. of the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition, Ischia, May 23-25, pp. 188-199, 2001.

[Karsai et al. 2003] Karsai G, Agrawal A, Shi F, On the Use of Graph Transformations for the Formal Specification of Model Interpreters, Journal of Universal Computer Science, Volume 9, Issue 11, pp. 1296-1321, November, 2003

[Kleppe et al. 2003] Kleppe A, Warmer J, Bast W, MDA Explained: The Model Driven Architecture-Practice and Promise, Addison-Wesley, 2003, ISBN: 032119442X

[Lédeczi et al. 2001] Lédeczi Á, Bakay Á, Maróti M, Völgyesi P, Nordstrom G, Sprinkle J, Karsai G, Composing Domain-Specific Design Environments. IEEE Computer 34(11), pp. 44-51, November, 2001

[Magyari et al. 2003] Magyari E, Bakay A, Lang A, Paka T, Vizhanyo A, Agrawal A, Karsai G, UDM: An Infrastructure for Implementing Domain-Specific Modeling Languages, The 3rd OOPSLA Workshop on Domain-Specific Modeling, OOPSLA 2003, Anahiem, California, October 26, 2003.

[Mellor et al. 2004] Mellor SJ, Scott K,Uhl A, Weise D, MDA Distilled - Principles of Model-Driven Architecture, Addison-Wesley, ISBN: 0201788918, 2004

[OMG MDA] OMG Model Driven Architecture homepage, www.omg.org/mda/

[OMG MDAGuide] MDA Guide Version 1.0.1, OMG, document number: omg/2003-06-01, 12th June 2003, www.omg.org/docs/omg/03-06-01.pdf

[OMG UML 2003]Unified Modeling Language Specification, v1.5, www.uml.org

[Pender &Pender 2003] Pender T, Pender T, UML Bible, Wiley, 2003, ISBN: 0764526049

[Pierce 1991] Pierce BC, Basic Categoy Theory for Computer Scientists, MIT Press, 1991, ISBN 0-262-66071-7

[Rozenberg 1997] Rozenberg G (ed.), Handbook on Graph Grammars and Computing by GraphTransformation: Foundations, Vol.1 World Scientific, Singapore, 1997.

[Samek 2002] Samek M, Practical Statecharts in C/C++,CMP Books, 2002, ISBN: 1578201101

[VMTS] http://avalon.aut.bme.hu/~tihamer/research/vmts/