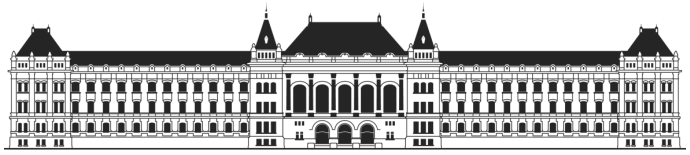

MŰEGYETEM 1782
Budapest University of Technology and Economics
Department of Measurement and Information Systems

Event-driven Model Transformations in Domain-specific Modeling Languages

PhD thesis booklet

István Ráth
MSc in technical informatics

Supervisor:
Dr. Dániel Varró, PhD
associate professor

Budapest, March 2011

Acknowledgements I wish to express my gratitude, first and foremost, to my supervisor, Dr. Dániel Varró. Without his continuous support, valuable insight, ideas, friendly advice and enthusiasm over the past six years, I would have never succeeded.

I am also grateful to my colleagues in the Fault Tolerant Systems Research Group, especially Prof. Dr. András Pataricza and Dr. István Majzik for their trust and support of my research. I highly acknowledge the VIATRA2 team, namely Dr. András Balogh, Gábor Bergmann, Ábel Hegedüs, Ákos Horváth, Zoltán Ujhelyi and Dr. Gergely Varró for the fruitful collaborative research work, and the fun we had. I am thankful to past members András Schmidt and Dávid Vágó for their contributions in the early beginnings. Many thanks to research associates Máté Kovács, Dr. Balázs Polgár, and Dániel Tóth, and my former MSc students: András Ökrös, Ádám Horváth, and all the others for their ideas, suggestions, contributions and hard work in using, testing and developing VIATRA2 over many years.

I would like to thank my colleagues at the University of Leicester, Prof. Dr. Reiko Heckel, Ajab Khan and Dr. Paolo Torrini for the fruitful collaborative work. I also highly acknowledge Philip Mayer at the Ludwig-Maximilians-Universität München for our joint work in the SENSORIA project. I wish to thank Dr. Krzysztof Czarnecki and his group for supporting my research visit at the University of Waterloo.

My work was supported by the SENSORIA (IST-3-016004, FP6), DIANA (AERO1-030985, FP6), SecureChange (ICT-FET-231101, FP7) and MOGENTES (ICT-216679, FP7) European research projects, and the PhD Candidate Scholarship of the Faculty of Electrical Engineering of the Budapest University of Technology and Economics.

Finally, I would like to thank my family and friends for their support and tolerance.

1 Introduction

1.1 Model-driven software engineering

Software engineering practice has several long-standing challenges, due to its inherent and ever increasing complexity. The mitigation of risks due to misunderstood requirements (which later evolve into expensive design errors, whose detection and fixing cost increases exponentially as development progresses) needs to be addressed by raising the level of abstraction to concepts that aid the direct involvement of domain experts in the development process. To achieve this, technology has to progress beyond today's error prone and ad-hoc practices to increase reuseability and traceability across the entire development toolchain and software lifecycle.

To answer these challenges, the *model-driven software development (MDSD)* paradigm is based on the idea that the developer should work with high abstraction level models during most of the development cycle, and the rest of the work should be assisted by automated tools, to reduce error-prone manual work. Source code should be generated to minimize the amount of business logic that is outside of the scope of modeling, and is only represented by handwritten code. Additionally, the abstract models should also be used to enhance the overall quality of the product, by supporting early design analysis using formal and semi-formal techniques [VP03] such as model checking, static analysis and model-based test generation.

Motivated to progress beyond UML's focus on documentation, the Object Management Group (OMG) issued the Model Driven Architecture (MDA) [MDA01] standard. MDA is essentially an approach to model-based software development utilizing OMG's flagship technologies, UML, the Meta Object Facility (MOF), XML Metadata Interchange (XMI), and the Common Warehouse Metamodel (CWM). MDA provides a template for model-driven development processes and summarizes best practices and design patterns.

MDA emphasizes the clear distinction between Platform Independent Models (PIM) and Platform Specific Models (PSM), thus, software development in MDA is envisioned as a three-step process. First, the Platform Independent Model is designed, which uses modeling concepts which are not platform specific. The second step is to generate a Platform Specific Model (PSM), which contains additional UML models, and represents a deployment/configuration of the system under design that is to run on the target platform. The transitions between PIM and PSM are facilitated using automated model transformation technology. Finally, application code is generated from the Platform Specific Model.

Model transformations in MDA Model transformation plays a key role in the overall process of MDA. The aim of model transformation is to carry out automated translations within and between modeling languages. By *horizontal model transformations*, information can be propagated within models on the same level of abstraction (e.g. model synchronization between PIM aspects), moreover, automated transformation tools also provide support for writing *vertical translations* that cross abstraction level boundaries (e.g. PIM-to-PSM mappings). These mappings are used in a wide spectrum of applications, e.g. for code generation, to map system models into mathematical domains for early analysis, various model management tasks such as model versioning, model migration, support for model-based test generation, model composition and well-formedness checking, and reverse engineering of source code into high level models.

Domain-Specific Modeling Despite the debate about the strengths and weaknesses of UML and MDA, industrial practice has proven that modeling can bring significant business advantages to those who master it [FC04]. One of the leading trends in model-centric development

today is the usage of *domain-specific modeling languages (DSMLs)* that are analogous to domain-specific languages (such as SQL) in the sense that in contrast to a general-purpose language, they are suited to express the notions of a focused problem domain much more precisely (on the expense that they are not well suited for usage in other domains). The origins of domain-specific languages can be traced back to the specialized programming languages of the 1960's and 70's, and the more recent initiatives of intentional programming and aspect-oriented programming, but the key emphasis in today's DSM technology is that modern tools are directed towards domain experts rather than programmers.

Thus, the main advantage of DSMLs over UML is the ability to capture domain knowledge concisely, at the right level of abstraction, so that domain experts can be directly involved in the development process. Moreover, domain-specific models have precise semantics (as their focus on the problem domain is tighter, such semantics are usually easier to specify than for general-purpose languages), and are thus well suited for automated processing and code generation. DSM is a top-down and vertical approach: instead of trying to create high abstraction level "interfaces" to source code, it gives the designer the freedom to use structures and logic that is specific to the target application domain, and thus, completely independent of programming language concepts and syntax. Early design analysis can also be more focused by exploiting domain-specific abstractions.

1.1.1 Domain-specific modeling languages in model-driven software engineering

In practical systems engineering, DSM concepts are frequently combined with other practices such as the MDA. A typical adaptation pattern is shown in Figure 1.

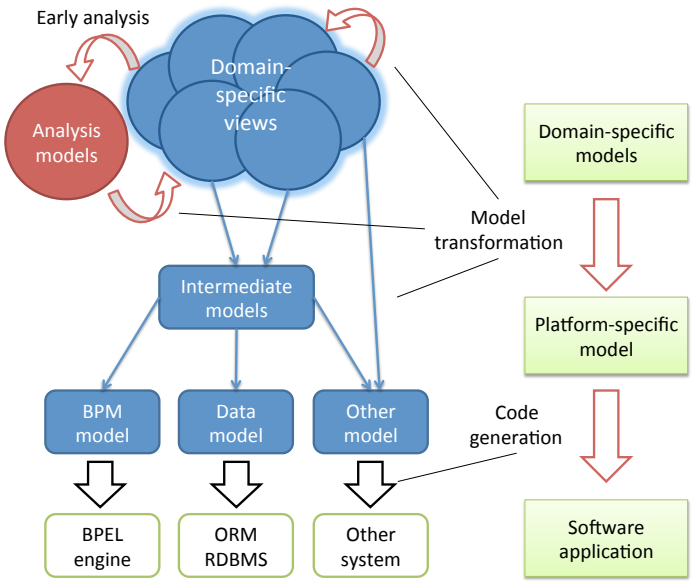

Figure 1: Domain-specific modeling and model transformations

Here, domain engineers capture the core constructs of the system under design by using visual design environments offering rich visual editors tailored to the specific application domain. These visual (and textual) languages specify various (inter-dependent) design aspects of the platform-independent model (PIM) and they may rely on standardized, general-purpose languages such as UML and UML profiles, and also on custom-made DSMLs specialized for the

needs of the organization or the concrete system-under design itself.

The information contained in this complex PIM is then further processed along the development chain. Instead of a single family of PSMs, various intermediate models may be used (which are frequently DSMs) that represent various stages and aspects of development (such as analysis models for early verification by formal methods, test models, deployment configurations, architecture prototype models, low-level models of source code to be generated etc). Finally, deployment artefacts and source code is generated from these low-level models.

1.1.2 Language engineering in development processes

Nowadays, modern development tools are increasingly used, aside from developing the target application itself, also for creating auxiliaries, plug-ins and generators that aid agile development and later re-use. Thus, while DSMLs provide great flexibility for rapid software development, the main challenge of their application is the difficulty and expensiveness of *language engineering* that becomes part of the software development process.

Even though nowadays there are plenty of ready-made DSMLs available (some of which have been recognized as international industry standards, such as SysML, AADL, AUTOSAR, and UML-MARTE in embedded software development, or BPEL, BPMN2 and SoaML in service-oriented business systems engineering, to name a few well-known examples), in many cases they are not a perfect fit for a particular software development problem. In such cases, DSMLs have to be developed or customized using *language engineering tools* such as the Eclipse Modeling Framework (EMF) [EMF] toolchain.

Such language engineering frameworks (e.g. [GME, LLMC04, Mic]) are used to create prototype modeling environments rapidly, and then to develop prototypes into fully featured development tools that can be integrated into complete toolchains. These tools, often integrated into development environments (such as Eclipse, in the case of EMF and its counterpart, the Graphical Modeling Framework [GMF]), have various degrees of support to traditional aspects of language engineering such as:

- *Concrete syntax*, to specify the human interface of the language (graphical or textual notation).
- *Abstract syntax* that maps the symbols of the language to the underlying (metamodeling) framework.
- *Mappings between representations* specify how abstract and concrete syntax models are interconnected and synchronized. The flexibility of the mapping technology has an elementary impact on the abstraction capabilities of the DSML, as it determines e.g. how much of the information contained in the abstract model can/has to be represented in the human-readable form.
- *Well-formedness rules*, which augment the abstract syntax to express more complicated, domain-specific constraints on the models.
- *Dynamic semantics (for simulation)* that specify behavioral aspects of DSML. Advanced environments, which support the in-design execution of such rules, enable designers to visualize the system-under-design as it is being executed (simulated).

1.2 Challenges

Unfortunately, despite the significant investment of research and development effort into state-of-the-art industrial language engineering tools (e.g. EMF has been around since 2003, with

numerous major releases), language engineering remains an expensive, time-consuming and many times cumbersome development task that requires special skills [DAR06].

Interestingly – and analogously to the situation with MDA –, one of the key areas, where *industrial* DSM technology also falls short, is the *lack of easy-to-use model transformations*. As a result, DSML-based development tools created with language engineering frameworks are *static and closed* in the following sense:

- While most tools have advanced support for writing code generators, it is **non-trivial to integrate models** into a library of other modeling languages – in many cases, ad-hoc technologies (operating on low-level XML representations) are used for this purpose. Such solutions are very hard to re-use.
- In many cases, they have **limited abstraction capabilities between abstract and concrete representations** (this is especially true of wide-spread graphical DSMLs based on the GMF); which limits the usability of graphical languages.
- Language **engineers have to learn complicated languages** such as OCL in order to specify even simple well-formedness constraints that cannot be expressed in the abstract syntax specification language (metamodel). This frequently leads to the omission of such constraints from end products, and has a serious impact on the quality of the tool.
- Many DSMLs are **restricted to static system modeling**, as it is challenging to create dynamic languages that incorporate behavioral aspects since the language prototyping tool rarely provides easy-to-use means to animate and/or simulate the models directly, within the modeling environment. While there are excellent dedicated tools for dynamic languages (e.g. MATLAB), in many cases they prove to be too expensive or complicated to use for a simple DSML prototype.

In current DSML tools, language engineering aspects related to model processing are covered by heterogeneous and isolated formalisms and mechanisms, which vary greatly from one tool to the other. Thus, customizable features or extensions depend either on the public programming interface or on the export-import facilities, and there is no end-to-end approach that covers integration aspects such as mappings between languages, model simulation support for dynamic analysis, or efficient evaluation of complex constraints. These are all hindering factors in DSML adaptation, since they considerably raise development costs.

In this thesis, I argue that model transformation technology can be adapted to the specific needs of domain-specific modeling language engineering, in order to provide support for advanced design aspects, which may significantly improve the capabilities of custom-built tools and reduce the required effort at the same time.

- **Challenge 1: Lack of a uniform approach to multi-domain modeling**. Domain-specific language engineering frameworks are focused on the development of a single language, or a family comprised of loosely connected languages. Mappings between identical or similar concepts (that may be present in multiple domains at the same time) are difficult to specify and maintain consistently. As most modeling environments are statically typed, model objects that represent information relevant in multiple domains (system model views) have to be replicated, which raises consistency and redundancy issues.
- **Challenge 2: Limitations of abstract and concrete syntax representation in domain-specific modeling language engineering frameworks**. State-of-the-art visual modeling

language engineering technology imposes limitations on how abstract and concrete syntax models are represented. In simplistic cases (e.g. MetaEdit+ [MEP]), these two model representations are connected by direct one-to-one links. In more advanced such as the GMF, abstract and concrete syntax models are stored in two distinct modeling layers, however there is a complicated synchronization mechanism between the two, limiting the ability of the designer to (i) create an arbitrary concrete syntax representation for a (pre-existing) language, or (ii) design a more complex abstract syntax for a given visual language. In practice, this reduces the scalability and usability of domain-specific modeling languages, especially as DSMLs are growing larger and more complex.

- **Challenge 3: Lack of integrated and generic support for the specification and execution of design-time, interactive simulation, parameterized by the behavioral semantics of the language**. Traditional DSM language engineering frameworks lack support for the specification and design-time execution of dynamic semantics. Even though many model transformation tools may be used as alternatives or complementary technologies to dedicated model simulation tools, they are not intrinsically targeted at language engineering, so they are typically used as complementaries to DSM frameworks. This raises integration issues which is ultimately a prohibiting factor in the rapid prototyping of dynamic languages.

- **Challenge 4: Lack of techniques to specify reactions to complex model changes in incremental transformations**. As DSMLs are interactive tools where the models are constantly changing, the adaptation of automated model transformation techniques to support advanced DSMLs are currently limited by a lack of support for the transparent detection and processing of model changes. While incremental model transformations (e.g. model synchronization) can be implemented using state-of-the-art tools, these only support the processing of elementary model changes (or heavily rely on extensive traceability models to calculate changes that are to be processed). Furthermore, the detection of changes relies on customized (user interface, or tool-specific) functionality, so generic change processing transformations are not feasible.

- **Challenge 5: No built-in support for incremental model synchronization and incremental code generation**. Even if complemented with state-of-the-art model transformations as auxiliary technology, language engineering environments lack facilities for the seamless integration of transformations to facilitate advanced features such as incremental, on-the-fly model synchronization or incremental code generation.

These research challenges have been motivated by practical challenges in tool integration, which is a key problem area in development projects that are carried out in heterogenous software environments. These problems are magnified in model-driven software development workflows, where a lot of semantic information needs to be propagated through a toolchain consisting of many complex domain-specific languages and tools.

2 Research method and new results

In order to reduce the costs of tool integration efforts, I have focused my research on the integrative applications of model transformations. The extension of VIATRA2 model transformations [VIA] to event-driven, efficient change processing (Thesis 1) opened up new possibilities for advanced language engineering features in domain-specific modeling languages (Thesis 2). These

advanced DSMLs and model transformation techniques could then be applied to practical applications in tool integration problems (Thesis 3).

2.1 Event-driven model transformations

As previous practical experience has shown, transformations used as a supporting infrastructure for advanced language engineering features required a novel execution approach, different from standard batch-style programs. Thus, I surveyed the literature on event-driven graph transformations and found that existing approaches [GdL07, MFF+06] of the time were limited by two assumptions: they relied on (i) special traceability models that represented an event (a model-specific notion of event), and (ii) external support from the tool environment to create these models when an event has been registered (an environment-specific detection of events).

In order to overcome these limitations, I elaborated the generalised concept of event-driven graph transformations, based on the incremental graph pattern matching approach [22] that has been developed at our research group during 2006-2007, and sharing motivational concepts with *live transformations* proposed by [HLR06]. As the central concept, I defined events as aggregated changes in the match set of a graph pattern, which is independent of modeling languages and the execution environment. I elaborated an *event-driven transformation language* based on the VIATRA2 language for graph triggers, and also proposed *execution semantics* [20]. The implementation of this approach became the live transformation engine of the VIATRA2 model transformation framework.

Thesis 1 I proposed a novel execution scheme for graph transformations (Challenge 4). *Live transformations* specify event-driven transformation rules, which, in contrast to batch-like traditional transformations, run continuously, and react to changes in the model on-the-fly.

1.1 I defined the *concepts for event-driven graph transformations based on incremental pattern matching* [20]. By this approach, changes of the model graph are detected by changing graph pattern match sets. Model transformation actions can make use of the *delta* of the match set, i.e. the exact specification of the model elements affected and the kind of change (creation, deletion, value update).

1.2 I defined *execution semantics for event-driven graph transformations* [7], based on event-condition-action formulas and a queue automaton model.

1.3 I elaborated a *graph transformation-based language for event-driven transformations* [20,7,4]), by adapting event-condition-action formulas. This specification language is based on the model transformation language of the VIATRA2 framework, and provides a formal approach for defining *graph triggers* as the basic units of event-driven live graph transformations.

1.4 Based on the transformation language and its execution semantics, I proposed *efficient implementation techniques and a software framework for event-driven graph transformations.* I verified the efficiency of the implementation using benchmarking techniques ([21,6]). The prototype implementation is part of the VIATRA2 framework ([17,7]).

Successive collaborative research ([4,2]) has extensively used these foundations. The implementation of the event-driven transformation framework was carried out jointly with András Ökrös, who was an MSc student under my supervision. The incremental graph pattern matcher is part of the PhD work of Gábor Bergmann.

The results of Thesis 1 are presented in Chapter 5 of the dissertation.

2.2 Automatic model transformations in domain-specific languages

Based on the feature survey that I had carried out [35] and my experience with existing language engineering frameworks for visual modeling languages around 2004-2005, I designed (in collaboration with Dávid Vágó) a novel language engineering framework for domain-specific visual modeling languages, called *ViatraDSM*. The main design guidelines of this framework were derived from the weaknesses observed in state-of-the-art approaches, and the approach specifically targeted two key challenges: (i) the complete separation of concrete and abstract syntax representations, generalizing the previously existing techniques [MFF+06]; (ii) high level support for advanced language-engineering aspects such as on-the-fly constraint evaluation and model simulation based on discrete event systems [SV08].

ViatraDSM was conceived to be a highly flexible and extensible stand-alone framework and a domain-specific front-end to the VIATRA2 modelspace. For this purpose, a meta-metamodel (or core metamodel) has been designed for both abstract and concrete syntax representations separately and a plugin-based infrastructure for language-specific extensions. Additionally, ViatraDSM supports graph transformations directly on domain-specific models using the underlying VIATRA2 engine. Similar architectural and metamodeling concepts have been later adapted for the Graphical Modeling Framework [GMF], the mainstream visual language engineering technology of Eclipse, with the exception of integrated transformation support.

Thesis 2 I developed a domain-specific modeling environment, which is based on the integrated model transformation approach. It provides high-level support for the specification of language design aspects (arbitrary abstract-concrete syntax synchronization, static well-formedness checking, model execution/animation and model translations) inside the modeling environment.

2.1 I developed *multi-aspect modeling for domain-specific languages* through a *uniform modelspace* [19] (Challenge 1), where both the meta- and instance models for multiple domain-specific modeling languages can be persisted, allowing for viewing the same instance models from different domain-specific perspectives. I also developed a common mapping meta-metamodel for visual domain-specific modeling languages, which allows flexible correspondence models between abstract and concrete syntax representations [29,7] (Challenge 2). These results are presented in Chapter 3.

2.2 I elaborated a technique for the *complete separation of abstract and concrete syntax of domain-specific modeling languages* (Challenge 2) based on generic event-driven model transformations [7,23,24]. These results are discussed in Chapter 6.

2.3 I developed a novel approach for the *design-time simulation of visual domain-specific models* [23,28,27,19] (Challenge 3) based on simulation rules specified by an enabledness condition (specified by a graph pattern) and an execution step (specified by graph transformation rules), to provide high-level support for debugging the dymanic semantics of executable domain-specific languages within the editing environment itself. These results are presented in Chapter 7.

2.4 I proposed *efficient implementation techniques* to all of the above results, which are available in the *ViatraDSM* tool [26]. I verified the *efficiency* of these implementation techniques using benchmarks [21,6,17,11], discussed in Chapter 7 of the dissertation.

The original, preliminary version of the DSM simulator in ViatraDSM (which provided the motivation for my work) has been developed by Dávid Vágó.

2.3 Tool integration based on change-driven transformations

Change-driven transformations. While graph transformation triggers provided a novel and straightforward technique of formalizing and executing live transformations, novel practical applications in tool integration necessitated to execute these transformations in an *asynchronous* manner, e.g. to propagate changes between models that are not directly accessible in the memory of the transformation engine (as formulated in Challenge 5). Thus, as an extension and generalization of the original event-driven approach, I developed the concept of *change-driven transformations* (CDTs) [15]. By this approach, events are represented by serialized *change history models*, and change processing can be executed independently of the actual occurrence of events. These techniques have been used for incremental code generation (change propagation to deployed process models) in the MOGENTES project (see details on tool integration work below). Additionally, CDTs also provide the basis for research work conducted in the SecureChange project.

Tool integration. As a contribution towards the practical and industrial application of my research results, I applied both the event-driven transformation technology and the domain-specific language engineering framework in several European Union research projects. In the context of the Sensoria EU FP6 project, I participated in the development of the Sensoria Development Environment (SDE) [3], a model-driven tool integration [KLN04] framework intended for the seamless integration of development and early analysis tools in the Eclipse environment. I developed the Viatra2 integration modules as an approach for parametrizing and invoking graph transformations through an abstract interface. I also designed and developed remote invocation support to allow the SDE to scale to distributed architectures; this feature was very important later on when we applied the SDE again to tool integration in the MOGENTES project.

In the MOGENTES project, we developed our tool integration technology further. I designed a modeling framework for the precise specification of tool integration workflows (in cooperation with several members of our research group), and applied the change-driven transformation concept to facilitate the on-the-fly manipulation of deployed workflow models to enable the rapid prototyping of tool integration processes. The ViatraDSM framework was used as a supporting environment for the workflow design toolkit.

In all, new applied research results in tool integration have been achieved based on all of my foundational research results (including domain-specific language prototypes, incremental transformations and change-driven transformations).

Thesis 3 I proposed the novel concept of *change-driven transformations* (CDTs), which operate on changes (represented as serialized change models or event objects) and decoupled host models, in order to allow asynchronous change propagation between non-materialized models. I applied this technology to non-intrusive incremental code generation, in the context of a novel model-based tool integration framework, where transformations can be used transparently as co-operating services. Information transfer between automated and semi-automated activities is facilitated using CDTs.

3.1 I developed the concept of *change-driven transformations* (CDTs) [15], which operate on changes (represented as serialized change models or event objects) and weakly referenced host models, in order to allow asynchronous change propagation between non-materialized models.

3.2 I proposed new implementation techniques and a software framework for event-driven and non-intrusive incremental code generation (Challenge 5), where changes are propa-

gated to non-materialized (deployed) models [4]. I elaborated a metamodeling framework for creating *change history reference models*, which represent atomic and complex changes applicable to both materialized and external host models.

3.3 I proposed efficient implementation techniques for remote service invocation in the SENSORIA Development Environment (SDE) [3], a tool integration framework developed within the context of the SENSORIA EU FP6 research project.

3.4 Based on the SDE, I proposed an extended implementation architecture for a model-based tool integration framework [16,1], where transformations can be used transparently as co-operating services to create transformation chains. Information transfer between automated and semiautomated activities is facilitated using change-driven model transformations.

3.5 I applied the incremental code generation approach to a custom domain-specific process description language. I elaborated a *model-based framework for capturing development workflows*, based on standardised modeling languages (SPEM, EPF, and jPDL) [16]. I developed code synchronization methods between high level process descriptions and a low-level executable jPDL representation deployed on the jBoss jBPM application server [15,4].

The results of theses 3.1, 3.2 and 3.5 are discussed in Chapter 8, while the results of 3.3 and 3.4 are discussed in Chapter 9 with Appendix C giving a detailed overview of the SDE.

The development of the SENSORIA Development Environment is a joint effort that has been coordinated by Philip Mayer at the Ludwig-Maximilians-Universität München. The remote service invocation feature has been developed in collaboration with Ádám Horváth, who was an MSc student under my supervision. Several members of the Fault Tolerant Systems Research Group contributed to extended tool integration framework for the MOGENTES project.

3 Applications of new results

In this section, I overview the practical applications of the results of my research.

3.1 The event-driven transformation engine of VIATRA2

The first version of the event-driven transformation engine of VIATRA2 has been developed in co-operation with two MSc students under my supervision, András Ökrös and Gábor Bergmann. Their work won the first prize at the Scientific Students' Association contest (Tudományos Diákköri Konferencia in Hungarian) of the Faculty of Electrical Engineering at the Budapest University of Technology and Economics in 2007, and again a first prize at the nationwide competition in 2009 (the report was co-supervised with Gergely Varró and Dániel Varró).

The results of this work are part of the official VIATRA2 Eclipse.org distribution as of Release 3 [VIA]. As such, it has been applied in various projects at our research group, ranging from research prototypes, tool integration (SENSORIA FP6 and MOGENTES FP7 EU projects [16]), and change impact analysis (in the SecureChange FP7 EU project [4]). This framework also provides the technological foundations to the newest version of ViatraDSM, as well as shares concepts and code with the constraint satisfaction solver engine [HV09] of VIATRA2 (which has been used in the DIANA EU FP6 project [HVS10]).

3.2 ViatraDSM

The results of my thesis have been implemented in ViatraDSM, an official add-on to the VIATRA2 release. A number of prototype domain-specific modeling languages (such as the tool integration scenario description language for the MOGENTES project, or multiple DSMLs for stochastic simulation at the University of Leicester) have been implemented using this tool.

3.3 Tool integration in the SENSORIA and MOGENTES EU research projects

The SENSORIA Development Environment (SDE) has been developed in co-operation with the project partners, especially the team lead by Philip Mayer at the Ludwig-Maximilians-Universität München. Our contributions (remote invocation support) have been developed in cooperation with Ádám Horváth, who was an MSc student under my supervision. Ádám has summarized his work in a Scientific Students' Association report (TDK), and won the third prize at the Faculity conference in 2008.

The SDE – and its modified and extended versions – have been used throughout the SENSORIA and later MOGENTES EU research projects by academic and industrial partners around Europe. This research has focused on generating configuration and deployment descriptions for web services based on high-level requirement models (SENSORIA) and automated test suite generation and test execution in embedded systems (MOGENTES).

3.4 Model simulation based on stochastic graph transformations

A collaborative research project between our research group and the group of Prof. Reiko Heckel at the University of Leicester has been started in 2009, as part of the SENSORIA project. The goal of this project is to develop a high performance stochastic graph transformation-based simulator, named VIATRA-GraTS [14]. The incremental event-driven technology elaborated in my thesis has been used in this tool. GraTS has since been applied to the simulation of peer-to-peer VoIP networks [12], and used in the MSc-PhD education programme at the University of Leicester.

3.5 EMF-IncQuery

In joint co-operative research targeting broader industrial applications of our technology, the VIATRA2 team has adapted the incremental pattern matcher ([22,21]) to be used on EMF models, a de-facto standard of model-based development tools. EMF-INCQuery [11] provides an efficient application platform for the results of all of my theses.

4 List of publications

Number of publications: 29
Number of peer-reviewed publications: 24
Approximate number of independent citations: 50

4.1 Book chapters (3)

[1] András Balogh, Gábor Bergmann, György Csertán, László Gönczy, Ákos Horváth, István Majzik, András Pataricza, Balázs Polgár, István Ráth, Dániel Varró, Gergely Varró. Workflow-driven tool integration using model transformations. In Gregor Engels, Claus Lewerentz,

Wilhelm Schaefer, Andy Schuerr, and Bernhard Westfechtel, editors, *Graph Transformations and Model-Driven Engineering*, volume 5765 of *Lecture Notes in Computer Science*, pages 224–248. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-17322-6_11.

[2] Gábor Bergmann, Artur Boronat, Reiko Heckel, Paolo Torrini, István Ráth, and Dániel Varró. Advances in model transformation by graph transformations: Specification, Analysis and Execution. In *Rigorous Software Engineering for Service-Oriented Systems - Results of the SENSORIA project on Software Engineering for Service-Oriented Computing*. Lecture Notes in Computer Science, vol. 6582. ISBN: 978-3-642-20400-5. Springer, 2011.

[3] István Ráth and Philip Mayer. The SENSORIA Development Environment. In *Rigorous Software Engineering for Service-Oriented Systems - Results of the SENSORIA project on Software Engineering for Service-Oriented Computing*. Lecture Notes in Computer Science, vol. 6582. ISBN: 978-3-642-20400-5. Springer, 2011.

4.2 Journal papers (5)

[4] Bergmann Gábor, István Ráth, Gergely Varró, and Dániel Varró. Change-driven model transformations: Change (in) the rule to rule the change. *Software and Systems Modeling*, 10:1–31, 2011. 10.1007/s10270-011-0197-9.

[5] Paolo Torrini, Reiko Heckel, István Ráth, and Gábor Bergmann. Stochastic graph transformation with regions. *Electronic Communications of the EASST, Proceedings of the Ninth International Workshop on Graph Transformation and Visual Modeling Techniques*, 2010.

[6] Ákos Horváth, Gábor Bergmann, István Ráth, and Dániel Varró. Experimental assessment of combining pattern matching strategies with VIATRA2. *International Journal on Software Tools for Technology Transfer (STTT)*, 12:211–230, 2010. 10.1007/s10009-010-0149-7.

[7] István Ráth, András Ökrös, and Dániel Varró. Synchronization of abstract and concrete syntax in domain-specific modeling languages. *Software and Systems Modeling*, 9:453–471, 2010. 10.1007/s10270-009-0122-7.

[8] Gábor Bergmann, István Ráth, and Dániel Varró. Parallelization of graph transformation based on incremental pattern matching. *Electronic Communications of the EASST, Proceedings of the Eighth International Workshop on Graph Transformation and Visual Modeling Techniques*, 18, 2009.

4.3 Conferences and workshops (21)

4.3.1 International conferences and workshops (17)

[9] Gábor Bergmann, Zoltán Ujhelyi, István Ráth, and Dániel Varró. A Graph Query Language for EMF models. *Theory and Practice of Model Transformations, Proceedings of the 4th International Conference on Model Transformations (ICMT)*, 2011. Accepted, acceptance rate: 25%.

[10] Ábel Hegedüs, Zoltán Ujhelyi, István Ráth, and Ákos Horváth. Visualization of traceability models with domain-specific layouting. In *Proceedings of the Fourth International Workshop on Graph-Based Tools*, 2010.

[11] Gábor Bergmann, Ákos Horváth, István Ráth, and Dániel Varró. Incremental evaluation of model queries over EMF models. In Dorina Petriu, Nicolas Rouquette, and Øystein Haugen, editors, *Model Driven Engineering Languages and Systems*, volume 6394 of *Lecture Notes in Computer Science*, pages 76–90. Springer Berlin / Heidelberg, 2010. Acceptance rate: 21%; DOI: 10.1007/978-3-642-16145-2_6.

[12] Ajab Khan, Reiko Heckel, Paolo Torrini, and István Ráth. Model-based stochastic simulation of P2P VoIP using graph transformation. In *Proceedings of the 17th International Conference on Analytical and Stochastic Modeling Techniques and Applications*, 2010.

[13] Ábel Hegedüs, Gábor Bergmann, István Ráth, and Dániel Varró. Back-annotation of simulation traces with change-driven model transformations. In *Proceedings of the Eighth International Conference on Software Engineering and Formal Methods*, pages 145–155, Pisa, 09/2010 2010. IEEE Computer Society, IEEE Computer Society. Acceptance rate: 22%.

[14] Paolo Torrini, Reiko Heckel, and István Ráth. Stochastic simulation of graph transformation systems. In David Rosenblum and Gabriele Taentzer, editors, *Fundamental Approaches to Software Engineering*, volume 6013 of *Lecture Notes in Computer Science*, pages 154–157. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-12029-9_11. Acceptance rate: 24%.

[15] István Ráth, Gergely Varró, and Dániel Varró. Change-driven model transformations. In Andy Schürr and Bran Selic, editors, *Model Driven Engineering Languages and Systems, 12th International Conference, MODELS 2009, Denver, CO, USA, October 4-9, 2009. Proceedings*, volume 5795 of *Lecture Notes in Computer Science*, pages 342–356. Springer, Springer, 2009. Springer Best Paper Award and ACM Distinguished Paper Award; Acceptance rate: 18%.

[16] Balázs Polgár, István Ráth, Zoltán Szatmári, and István Majzik. Model-based Integration, Execution and Certification of Development Tool-chains. In *2nd ECMDA Workshop on Model-Driven Tool and Process Integration*, 2009.

[17] Gábor Bergmann, Ákos Horváth, István Ráth, and Dániel Varró. Efficient model transformations by combining pattern matching strategies. In Richard Paige, editor, *Theory and Practice of Model Transformations*, volume 5563 of *Lecture Notes in Computer Science*, pages 20–34. Springer Berlin / Heidelberg, 2009. Acceptance rate: 23%; DOI: 10.1007/978-3-642-02408-5_3.

[18] András Balogh, András Pataricza, and István Ráth. Automated verification and validation of domain specific languages and their applications. In *Proceedings of the 4th World Congress for Software Quality*, pages 1–6, Bethesda, USA, 2009.

[19] István Ráth, Dávid Vágó, and Dániel Varró. Design-time simulation of domain-specific models by incremental pattern matching. In *IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2008, Herrsching am Ammersee, Germany, 15-19 September 2008, Proceedings*, pages 219–222. IEEE, 2008. Acceptance rate: 29%.

[20] István Ráth, Gábor Bergmann, András Ökrös, and Dániel Varró. Live model transformations driven by incremental pattern matching. In Antonio Vallecillo, Jeff Gray, and Alfonso Pierantonio, editors, *Proc. First International Conference on the Theory and Practice of Model Transformations (ICMT 2008)*, volume 5063/2008 of *Lecture Notes in Computer Science*, page 107–121. Springer Berlin / Heidelberg, Springer Berlin / Heidelberg, 2008. Acceptance rate: 31%.

[21] Gábor Bergmann, Ákos Horváth, István Ráth, and Dániel Varró. A benchmark evaluation of incremental pattern matching in graph transformation. In Hartmut Ehrig, Reiko Heckel, Grzegorz Rozenberg, and Gabriele Taentzer, editors, *Graph Transformations*, volume 5214 of *Lecture Notes in Computer Science*, pages 396–410. Springer Berlin / Heidelberg, 2008. Acceptance rate: 40%; DOI: 10.1007/978-3-540-87405-8_27.

[22] Gábor Bergmann, András Ökrös, István Ráth, Dániel Varró, and Gergely Varró. Incremental pattern matching in the VIATRA transformation system. In *GRaMoT'08, 3rd International Workshop on Graph and Model Transformation.* 30th International Conference on Software Engineering, 2008.

[23] István Ráth and Dániel Varró. Challenges for advanced domain-specific modeling frameworks. In *International Workshop on Domain Specific Program Development (DSPD 2006)*, Nantes, France, July 2006.

[24] István Ráth. Declarative mapping between abstract and concrete syntax of domain-specific visual languages. In *The Proceedings of the Fifth Conference of PhD Students in Computer Science*, 2006.

[25] András Balogh, Attila Németh, András Schmidt, István Ráth, Dávid Vágó, Dániel Varró, and András Pataricza. The VIATRA2 model transformation framework. *Tool demo at the First European Conference on Model Driven Architecture - Foundations and Applications*, 2005.

4.3.2 National conferences (4)

[26] István Ráth. Modelltranszformációk integrált alkalmazása domain-specifikus nyelvekben. In *Tavaszi Szél Konferenciakiadvány*, 2009.

[27] István Ráth. Enhancing design-time model execution in domain-specific languages by incremental pattern matching. In *Proceedings of the 16th PhD Minisymposium*, pages 16–20. Budapest University of Technology and Economics, Department of Measurement and Information Systems, 2009.

[28] István Ráth. Design-time simulation of domain-specific modeling languages by interactive model transformation. In *Proceedings of the 15th PhD Minisymposium*, pages 58–62. Budapest University of Technology and Economics, Department of Measurement and Information Systems, 2008.

[29] István Ráth. Challenges for advanced domain-specific modeling frameworks. In *Proceedings of the 14th PhD Minisymposium*, pages 118–120. Budapest University of Technology and Economics, Department of Measurement and Information Systems, 2007.

4.4 Other (6)

4.4.1 Technical reports (4)

[30] Budapest University of Technology and Economics. With contributions by István Ráth. Report on the Framework Implementation. *MOGENTES Project Deliverables D2.2.a, b, c* June 2009, December 2009, March 2011.

[31] Ábel Hegedüs, István Ráth, Dániel Varró. From BPEL to SAL and Back: a Tool Demo on Back-Annotation with VIATRA2. *SEFM Posters and Tool Demo Session Track*, 09/2010. pp. 35–42. ISBN: 978-88-7958-006-9.

[32] Ákos Horváth, Dénes Monostori, András Balogh, Imre Kocsis, Gergely Pintér, Antal Fazakas, István Ráth, Dániel Varró, Ivo Viglietti, Massimo Cifaldi and Tobias Schoofs. Report on the definition of the AIDA development means. *DIANA Project Deliverable DC3.1* April 20, 2010.

[33] Philip Mayer, István Ráth, Ádám Horváth. Report on the SENSORIA Development Environment. SENSORIA *Project Deliverables D7.4.c–d.* August 31, 2008 – January 13, 2010.

4.4.2 Scientific Students' Association Report

[34] András Schmidt, István Ráth, Dávid Vágó. Automated model transformations in domain specific visual languages. *Scientific Students' Associations Report*, Budapest University of Technology and Economics, 2005. First Prize at the BUTE Conference of the Electrical Engineering and Informatics Faculty (2005), and First Prize at the National Conference (2007).

4.4.3 Master's thesis

[35] István Ráth. Declarative specification of domain specific visual languages. Master's thesis, Budapest University of Technology and Economics, 2006.

References

[DAR06] Hanna Farah Daniel Amyot and Jean-François Roy. Evaluation of Development Tools for Domain-Specific Modeling Languages. *System Analysis and Modeling: Language Profiles, Springer LNCS*, 4320/2006:183–197, December 2006. DOI 10.1007/11951148_12.

[EMF] *Eclipse Modeling Framework*. `http://www.eclipse.org/emf`.

[FC04] David S. Frankel and Steve Cook. Domain-specific modeling and model driven architecture. *MDA Journal*, 2004. `http://www.bptrends.com/publicationfiles/01-04COLDomSpecModelingFrankel-Cook.pdf`.

[GdL07] Esther Guerra and Juan de Lara. Event-driven grammars: Relating abstract and concrete levels of visual languages. *Software and Systems Modeling*, 6(3):317–347, 2007.

[GME] GME. The Generic Modeling Environment. `http://www.isis.vanderbilt.edu/Projects/gme`.

[GMF] Eclipse Graphical Modeling Framework. `http://www.eclipse.org/gmf`.

[HLR06] David Hearnden, Michael Lawley, and Kerry Raymond. Incremental Model Transformation for the Evolution of Model-Driven Systems. In *Proc. of 9th International Conference on Model Driven Engineering Languages and Systems (MODELS 2006)*, volume 4199 of *LNCS*, pages 321–335, Heidelberg, Germany, 2006. Springer Berlin.

[HV09] Ákos Horváth and Dániel Varró. CSP(M): Constraint satisfaction problem over models. In Andy Schürr and Bran Selic, editors, *Model Driven Engineering Languages and Systems, 12th International Conference, MODELS 2009, Denver, CO, USA, October 4-9, 2009. Proceedings*, volume 5795 of *Lecture Notes in Computer Science*, pages 107–121. Springer, Springer, 2009. Acceptance rate: 18%.

[HVS10] Ákos Horváth, Dániel Varró, and Tobias Schoofs. Model-driven development of ARINC 653 configuration tables. In *29th IEEE & AIAA Digital Avionics System Conference (DASC)*, Salt Lake City, US, 10/2010 2010. IEEE, IEEE.

[KLN04] Gabor Karsai, Andras Lang, and Sandeep Neema. Design patterns for open tool integration. *Software and Systems Modeling*, 4(2):157–170, 2004.

[LLMC04] Tihamér Levendovszky, László Lengyel, Gergely Mezei, and Hassan Charaf. A systematic approach to metamodeling environments and model transformation systems in VMTS. In *Proc. GraBaTs 2004: International Workshop on Graph Based Tools*. Elsevier, 2004.

[MDA01] *The Object Management Group: Model Driven Architecture — A Technical Perspective*, September 2001. `http://www.omg.org`.

[MEP] MetaCase MetaEdit+. `http://www.metacase.com/mep/`.

[MFF+06] Pierre-Alain Muller, Franck Fleurey, Frédéric Fondement, Michel Hassenforder, Rémi Schneckenburger, Sébastien Gérard, and Jean-Marc Jézéquel. Model-Driven Analysis and Synthesis of Concrete Syntax. *Model Driven Engineering Languages and Systems, Springer LNCS*, 4199/2006:98–110, November 2006. DOI 10.1007/11880240_8.

[Mic] Microsoft. DSL Tools. `http://lab.msdn.microsoft.com/teamsystem/workshop/dsltools/default.aspx`.

[SV08] Eugene Syriani and Hans Vangheluwe. Programmed graph rewriting with DEVS. In Andy Schürr, Manfred Nagl, and Albert Zündorf, editors, *Applications of Graph Transformations with Industrial Relevance*, volume 5088 of *Lecture Notes in Computer Science*, pages 136–151. Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-89020-1_11.

[VIA] *VIATRA2 Framework*. An Eclipse GMT Subproject (`http://www.eclipse.org/gmt/`).

[VP03] Dániel Varró and András Pataricza. VPM: A visual, precise and multilevel metamodeling framework for describing mathematical domains and UML. *Journal of Software and Systems Modeling*, 2(3):187–210, October 2003.