Scalability in System Design and Management, the MONDO Approach in an Industrial Project

Alessandra Bagnato[1], Etienne Brosse[1], Marcos Aurélio Almeida da Silva[1], Andrey Sadovykh[1]

[1] R&D Department, SOFTEAM, 9 Parc Ariane, Guyancourt, France
alessandra.bagnato@softeam.fr, etienne.brosse@softeam.fr, marcos.almeida@softeam.fr, andrey.sadovykh@softeam.fr

Abstract. The current system designs and management technologies are being stressed to their limits in terms of collaborative development, efficient management and persistence of large and complex models. As such, a new line of research is imperative in order to achieve scalability across the system design space. Scalability in system design has different dimensions: domains, team localizations, number of engineers, size and management of the engineering artefacts, interoperability and complexity of languages used. This paper depicts how the MONDO FP7 EU project (http://www.mondo-project.org/) aims to comprehensively tackle the challenge of scalability in system design and management by developing the theoretical foundations and an open-source implementation of a platform for scalable modelling and model management. An industrial case study is also presented. The system designed in this case study is distributed among several and dependent units, domains, and languages.

1 Introduction

As Model Driven Engineering (MDE) is increasingly applied to larger and more complex systems, the current generation of modelling and model management technologies have being pushed to their limits in terms of capacity and efficiency. Therefore additional research is imperative in order to enable MDE to keep up with industrial practice and continue delivering its widely recognized productivity, quality, and maintainability benefits.

In the following section, we will present how the MONDO project plans to handle the increasingly important challenge of scalability in MDE. In section 3, we present how some issue tackled by the MONDO [2] approach have already been implemented and used on an industrial project within the Modelio Modeling tool environment [1].

2 MONDO Approach

Achieving scalability in modelling and MDE involves being able to construct large models and domain-specific languages in a systematic manner, enabling teams of modelers to collaboratively construct and refine large models, advancing the state-of-the-art in model querying and transformation tools so that they can cope with large models (of the scale of millions of model elements), and providing an infrastructure for efficient storage, indexing and retrieval of large models. To address these challenges, MONDO will develop or optimize algorithms at different levels of the system modelling. Obviously techniques and tools will be implemented at model and model engineering levels. MONDO approach also takes into account at higher levels, depicted in **Fig. 1**, i.e. the meta-model engineering and meta-model levels.

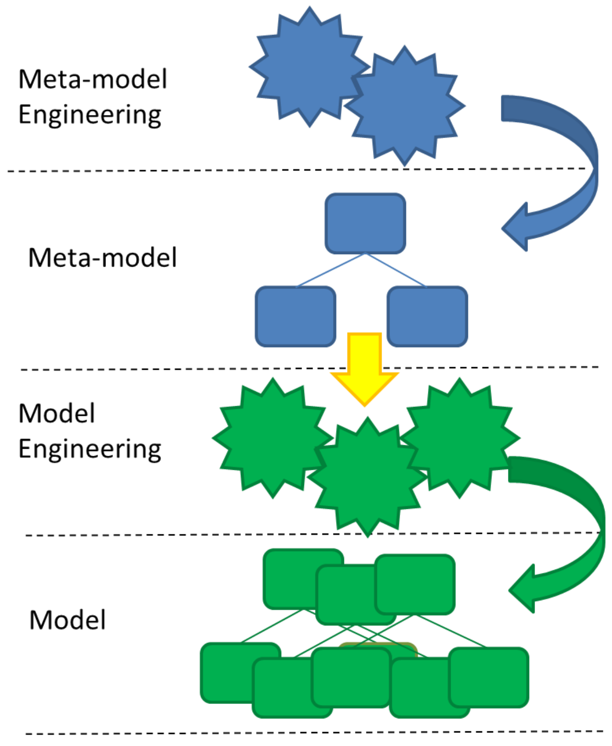


Fig. 1. The levels considered by the MONDO approach

In section 2.1, we describe the MONDO research field at the meta-model and meta-model engineering levels. Section 2.2 presents the MONDO targeted work at the model and model engineering levels. Finally sections 2.3 and 2.4 highlight that the techniques explored apply at multiple levels in the MONDO approach.

2.1 Scalability at the meta-model and meta-model engineering levels

The first objective of MONDO is to develop scalable techniques and processes for the engineering meta-model modelling or domain-specific languages (DSLs)[9]. Providing techniques for the efficient support and optimization of languages for which large models (instance of these meta-model or DSL) are expected. Also, the complexity or size concerns are in the definition of the languages themselves and hence we will propose methods; enable the engineering of such languages.
MONDO will complement these techniques with processes facilitating their use in complex projects dealing with large artefacts. In order to achieve these objectives, MONDO will provide techniques for developing languages enabling scalable modelling, techniques for developing languages enabling scalable modelling, scalable concrete syntaxes, scalable MDE processes. The following paragraphs and more in detail in [9] describe the technologies supported by the project.

Techniques for developing languages enabling scalable modelling.

MONDO will investigate techniques to construct an efficient infrastructure for DSLs that are expected to have large instance models. In this way, it will propose techniques to describe optimizations in the storage/retrieval of large instance models. These optimizations will be specified at the meta-model level, when the DSL is being built, and could be suggested by a recommender system. We will provide support for defining and reusing abstractions for DSMLs that will produce simpler views of large models. Another line of work will be the automated modularity support for DSLs, as well as the possibility to build large models by compositing reusable model fragments and template models from a library spanning across heterogenous technical spaces (DSLs, UML, Matlab/Simulink, etc).

Techniques for developing languages enabling scalable modelling.

Often, the definition of a DSL is itself complex, which renders its construction in an ad-hoc manner challenging. Therefore, we will provide methods and techniques to construct meta-models for large, complex DSLs. In particular, we aim to develop techniques for incremental, example-based construction of meta-models, supported by automated guidelines or meta-model patterns. We will also develop methods for meta-model construction by reusing existing fragments and meta-model templates from a library spanning across heterogenous technical spaces (DSLs, UML, XML schemas, etc).

Scalable concrete syntaxes.

MONDO will develop techniques for designing scalable visual concrete syntaxes. These techniques should enable the visualization and navigation of large models, including their visualization at different levels of detail or abstraction, in connection with the defined abstractions at the abstract syntax level. Techniques will also be developed to automate the visualization and exploration of large scale models, for which no concrete graphical syntax has been defined. The techniques developed in this task will be delivered as ready-to-use library elements, enabling the rapid development of visual editors with built-in support for abstractions and facilities for model exploration.

Scalable MDE processes.

MONDO will investigate processes (and tool support for them) for the use of MDE in complex projects, dealing with large artefacts. Hence, we will bring ideas from agile development into MDE. For example, similar to continuous integration, we will verify the consistency of models on the server in the same way we run automated builds and unit tests. We will also apply "by-example" techniques to the incremental construction of large, complex DSLs, and adopt techniques and processes from reutilization-based development in order to characterize reusable assets and facilitate their reutilization.

2.2 Scalability of the model operations

Any non-trivial MDE project involves querying and manipulation of a substantial number of models. These model manipulation operations are usually implemented as model-to-model transformations that take as input one or more source models and generate as output one or more target models, where target and source models can conform to the same or to different meta-models. Model queries are primary means to extract views and to formally capture and validate well-formedness constraints, design rules and guidelines on the fly. Therefore, scalability of model queries and transformations is a key element in any scalable MDE solution. Our experience with industrial case studies is that current transformation technologies do not scale, which discourages some potential adopters from using MDE.

So the second objective of MONDO is to create a new generation of model querying and transformation technologies that can solve this problem. We propose to build a reactive transformation engine by combining incremental change propagation with lazy computation. We plan to provide the engine with strong parallelization properties, to be able to fully exploit distributed/cloud execution environments.
Re-evaluating a validation query or regenerating a full target model after a few local changes in the underlying (source) model can be particularly inefficient. An incremental query evaluation technique will propagate model changes directly to the affected queries to incrementally update their result set. An incremental transformation algorithm will minimize the number of transformation rules to be re-executed according to the changes on the source model, while ensuring the synchronicity between source and target models.

Lazy / on-the-fly / on-demand creation of target models.

Generating a full target model from a source model can be time consuming, especially since often only parts (in proportion to the full model size) of the target model will be accessed. A lazy evaluation algorithm would try to execute the transformation on-demand, only producing the subsets of the target model when they actually need to be accessed.

Queries and Transformations in the cloud: parallel/distributed execution Model queries and transformations can benefit from leveraging distributed cloud architecture for their execution. If adapted, queries and transformations can rely on the scalability of cloud architectures to deal with large models. Two kinds of adaptations can be performed: 1) partitioning of the model and parallel execution of the full transformation on each model slice and 2) partitioning of the transformation and execution of transformation subsets on different cloud nodes. In both cases the result must be merged at the end.

Infinite/streaming transformations.

In some MDE application scenarios we need to deal with infinite models (there can be a continuous influx of data into the model). Therefore, a transformation cannot wait until the model is complete to start transforming it; instead it needs to be able to output target model elements as source elements are becoming available. Traces between target and source elements have to be kept in memory until the system reaches a certain degree of confidence that they will not be needed for future computations.

Integration with scalable persistence mechanisms.

Here MONDO will develop interfaces that enable model management languages to query and modify models persisted using the technologies proposed in an efficient manner.

2.3 Collaborative modelling

The objective at this level is to provide new collaborative modeling tools for geographically distributed teams of engineers working on large-scale models using heterogeneous devices (desktop computers, laptops, tablets, mobile phones, etc). These collaborative modeling tools will allow multiple teams from different stakeholders (such as system integrators, subcontractors, certification bodies) to simultaneously access the server-side model as clients in a scalable and secure way respecting access control policies.

For this purpose, MONDO will first investigate and adapt offline (asynchronous, long transaction) and online (synchronous, short transaction) collaboration patterns for models. Offline collaboration (like SVN or CVS) is widely used in collaborative software engineering where developers commit a larger portion of changes as a long transaction (e.g. at the end of the day). Online collaboration is popular in collaborative document authoring (analogously to GoogleDocs or LucidCharts) where a group of collaborators cooperatively and simultaneously edit the same document, and each individual change is immediately committed to a master copy.

For offline collaborative modeling, novel intelligent, user-guided techniques will be developed based upon design space exploration techniques to resolve conflicts upon concurrent commits. Furthermore, the work package will investigate how to determine the proper size and boundaries of model fragments to balance collaboration and performance of underlying queries and transformations. Intelligent, view-driven dynamic locking mechanisms will be proposed based upon incremental model queries to avoid unintentional concurrent changes of model elements, which is a key issue for both online and offline collaborative modeling. Secure model access control policies will be defined uniformly for online and offline collaboration, and will be enforced to facilitate collaboration between teams of different stakeholders.

A multi-device collaborative modeling framework will be developed to demonstrate the feasibility and scalability of different collaboration patterns over heterogeneous devices as collaboration means. An interface will be defined to encapsulate this collaborative modeling layer and will be offered as services to high-level domain-specific modeling, query and transformation tools. More details on the planned work on collaborative modeling can be found at [3] [4] [5] [6].

Primitives and Patterns for Collaborative Modeling.

Various collaboration primitives and patterns for offline (e.g. SVN-based) and online strategies (e.g. collaborative modeling authoring sessions) will be investigated. This task also aims to provide consistency management techniques for collaborative modeling adapted from version control systems including locking, transaction handling with commit, conflict management and resolution. Furthermore, this task also incorporates the development of a new version of a model by a team of developers during collaborative modeling sessions where the consistency of persistent storage can be temporarily violated to improve collaboration.

Secure Access Control for Collaborative Modeling.

Confidentiality support for collaborative modeling by providing secure access control will be also provided. Access control policies will be defined at a high level uniformly for online and offline cases, which will be mapped to the security means provided by the underlying model storage frameworks.

Interface for Collaborative Modeling.

At this point MONDO will define an interface for collaborative modeling services (including update, commit, lock, etc.), which will by used by advanced modeling and transformation tools. This interface will be aligned with underlying scalable model storage mechanisms, and will offer means for both online and offline collaboration.

Multi-Device Collaborative Modeling Tool.

A multi-device collaborative modeling tool will be developed which allows various development teams to simultaneously and collaboratively access, query and manipulate the underlying models using heterogeneous devices as a collaboration means. Integration with existing domain-specific modeling tools and the persistence layer will be provided

2.4 Efficient model persistence

The aim of this level is to develop an efficient model persistence format in order to address the limitations of the current standards (e.g. XMI), and to deliver a scalable model indexing framework. This will enable the querying and transformation languages and the collaborative modelling tools developed in to efficiently manage large and heterogeneous models.

3 Industrial Case study

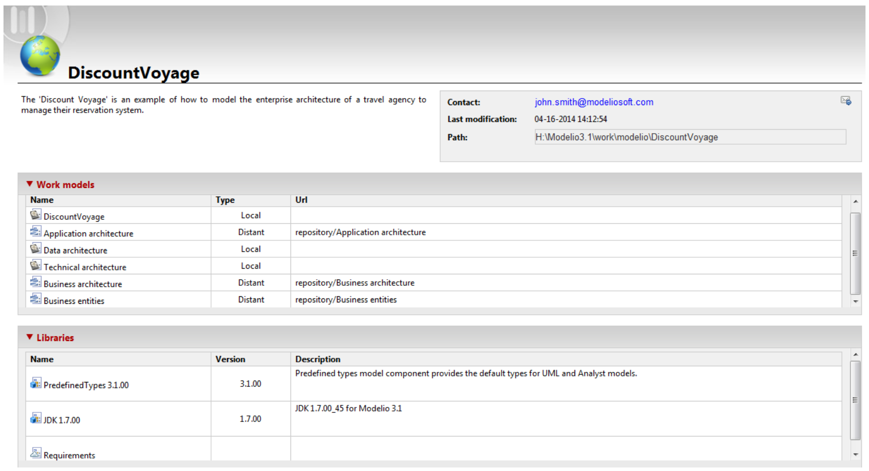
Not all points developed in the MONDO approach (cf. Section 2 MONDO Approach) have been implemented or even specified yet within the project being the project in its early stage. We detail in the following section the lazy loading of models solution that already exist within our Industrial Case Study and it is currently implemented within the Modelio Modeling environment [1] and that will serve as a case study to evaluate the MONDO project results.
The case study consists of a distributed modelling of a system between a heterogeneous set of users. For this we used on one hand the distributed framework - described in section 3.1 Modelio Distributed framework - and on the other one the view/viewpoint and lazy loading mechanism both of them provided by Modelio tool [1]. All the technologies provided by MONDO will be evaluated within the same industrial case study.

3.1 Modelio Distributed framework

The concept of the "modeling project" has been completely overhauled under Modelio[1]. A Modelio project now groups a certain number of local or remote models. Local models, or working models, can be edited by the user, while remote models, which are accessed by HTTP, are used to integrate model libraries published by other contributors into the project. For example **Fig. 2** shows our case study configured with:

- Three local models named “DiscountVoyage”, “Data architecture”, and “Technical architecture”
- Three remote modes names “Application architecture”, “Businnes architecture”, and “Businnes entities”.
- Two locals libraries i.e. “PredefinedTypes 3.1.00” and “JDK 1.7.00”
- One remote library i.e. “Requirements”.

In practical terms, this means that once the project configuration has been established, the different models are viewed as a single model in the model browser as depicted in **Fig. 3**. They can be used transparently for modeling work.



DiscountVoyage

The 'Discount Voyage' is an example of how to model the enterprise architecture of a travel agency to manage their reservation system.

Contact:	john.smith@modeliosoft.com
Last modification:	04-16-2014 14:12:54
Path:	H:\Modelio3.1\work\modelio\DiscountVoyage

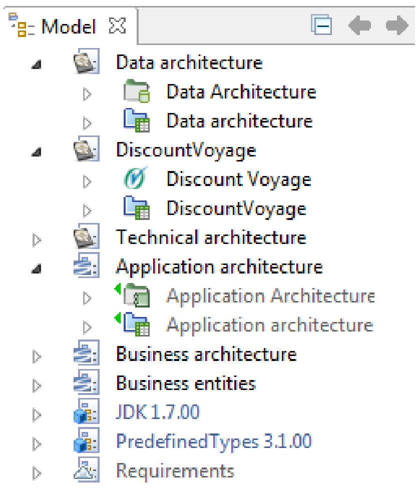
Work models

Name	Type	Url
DiscountVoyage	Local	
Application architecture	Distant	repository/Application architecture
Data architecture	Local	
Technical architecture	Local	
Business architecture	Distant	repository/Business architecture
Business entities	Distant	repository/Business entities

Libraries

Name	Version	Description
PredefinedTypes 3.1.00	3.1.00	Predefined types model component provides the default types for UML and Analyst models.
JDK 1.7.00	1.7.00	JDK 1.7.00_45 for Modelio 3.1
Requirements		

Fig. 2. Distributed framework configuration



Model

- Data architecture
  - Data Architecture
  - Data architecture
- DiscountVoyage
  - Discount Voyage
  - DiscountVoyage
- Technical architecture
- Application architecture
  - Application Architecture
  - Application architecture
- Business architecture
- Business entities
- JDK 1.7.00
- PredefinedTypes 3.1.00
- Requirements

Fig. 3. Distributed framework rendering

The models combination used of each project depends of your set of concerns and of course theirs availabilities regarding the defined access right. This flexibility in term of model management allows the creation of different community or groups sharing set of models.

When you deal with accessibility configuration, you also have to deal with the object “absence”. “Absent” objects are elements which are momentarily inaccessible but still known by their identifier and their type. In Modelio, absent objects are represented in simplified form (name, type icon) and therefore do not prevent the model from being used. Links modeled towards an “absent” object are retained. When an “absent” object is re-established and becomes accessible once again, these links are automatically re-established in their complete definition.

This mechanism enables, for example, a Modelio project open in the modeler to « survive » a network disconnection which deprives it of remote libraries and the elements they contain.

3.2 View, Viewpoint and lazy loading.

The SysML modelling standard defines a general-purpose modeling language for systems engineering applications, called the OMG Systems Modeling Language (OMG SysML™) [8]. Throughout the rest of the paper, the language will be referred to as SysML. SysML supports the specification, analysis, design, verification, and validation of a broad range of complex systems. These systems may include hardware, software, information, processes, personnel, and facilities.
The Open Group Architecture Framework, or TOGAF [7] modelling standard, is the de facto global standard for Enterprise Architecture. TOGAF provides the methods and tools for assisting in the acceptance, production, use, and maintenance of enterprise architecture. It is based on an iterative process model supported by best practices and a re-usable set of existing architecture assets.

Both these standards and others ones, define the view and viewpoint concepts. In short, a view is a portion of your system i.e. a representation of your whole system from the perspective of a related viewpoint. A viewpoint is a particular set of concerns. The implementation of these two concepts under Modelio allows, according to a specific set of concerns, a reduction of manipulated element. For example **Fig. 5** depicts the Modelio full viewpoint where many elements are shown and necessarily loaded in memory. Contrary to the Modelio trace viewpoint, cf.

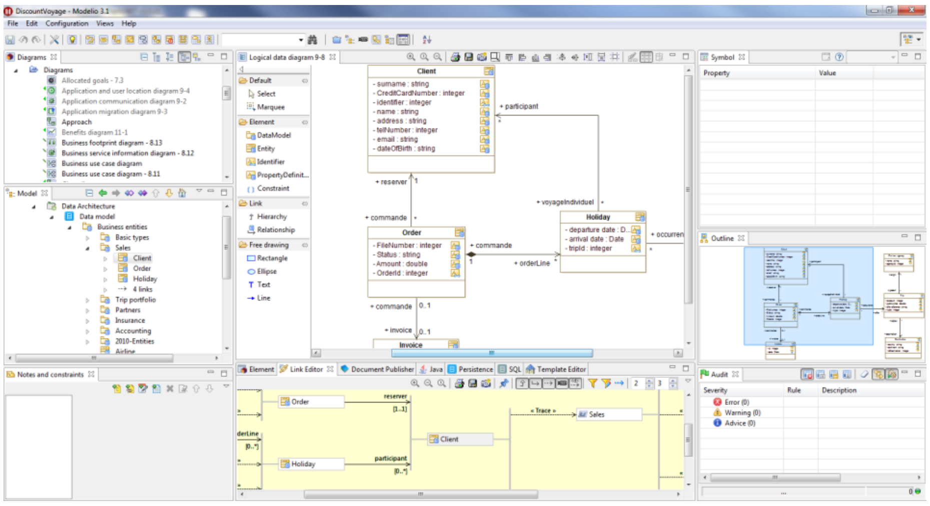


Fig. 4, focusses to the links from a given element but not to other relative concept. The association of this view/viewpoint mechanism and a lazy loading is able to minimize as much as possible the number of element loaded in memory and consequently the size of allocated memory.

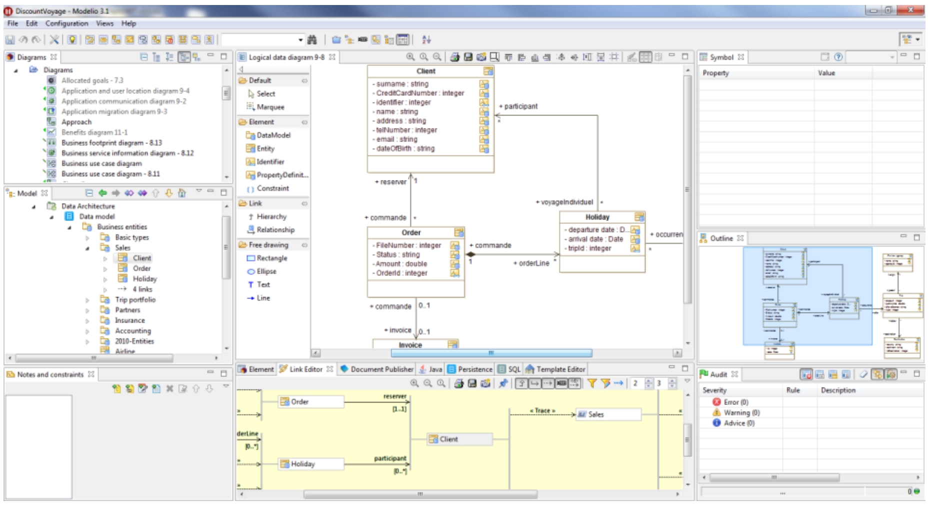


Fig. 4. Modelio full viewpoint

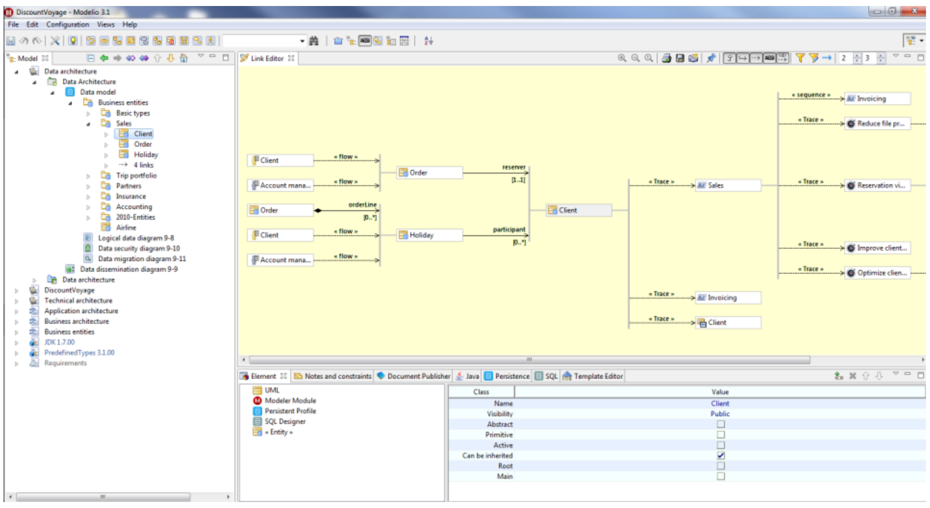


Fig. 5. Modelio trace viewpoint

4 Conclusion

As we have already stated the MONDO project is currently at its early stage so most optimization attempts must be specified, implemented and of course evaluated on relevant and industrial system design cases. But we are optimistic about expected results mainly because some techniques already present in current tools and presented in this paper have shown good results and will improve as the project advances further.

Acknowledgements

The research leading to these results is partially supported from the European Community Seventh Framework Programme (FP7/2007-2013) under grant agreement no FP7-611125. We would also like to acknowledge all the members of the MONDO (Scalable Modelling and Model Management on the Cloud) Consortium for their valuable help.

References

1. "Modelio," [Online]. Available: http://www.modeliosoft.com [Accessed 25

June 2014].

2. "MONDO Project Web Site" [Online]. Available: http://www.mondo-project.org/. [Accessed 25 June 2014].

3. Ujhelyi, Z., Bergmann, G., Hegedüs, Á., Horváth, Á., Izsó, B., Ráth, I., Szatmári, Z., and Varró, D., "EMF-IncQuery: An Integrated Development Environment for Live Model Queries", Science of Computer Programming, 2014.

4. Ujhelyi, Z., Horváth, Á., Varró, D., Csiszár, N I., Szőke, G., Vidács, L., and Ferenc, R., "Anti-pattern Detection with Model Queries: A Comparison of Approaches", IEEE CSMR-WCRE 2014 Software Evolution Week: IEEE, 02/2014.

5. Szárnyas, G., Izsó, B., Ráth, I., Harmath, D., Bergmann, G., and Varró, D., "IncQuery-D: A Distributed Incremental Model Query Framework in the Cloud", ACM/IEEE 17th International Conference on Model Driven Engineering Languages and Systems, MODELS 2014, Valencia, Spain, Springer, 2014.

6. Dávid, I., Ráth, I., and Varró, D., "Streaming Model Transformations By Complex Event Processing", ACM/IEEE 17th International Conference on Model Driven Engineering Languages and Systems, MODELS 2014, Valencia, Spain, Springer, 2014.

7. "TOGAF® Version 9.1 Enterprise Edition Web Site" [Online]. Available: http://www.opengroup.org/togaf/ [Accessed 25 June 2014].

8. "SysML Version 1.2 Web Site" [Online]. Available: http://www.omg.org/spec/SysML/1.2 [Accessed 25 June 2014].
9. Dimitrios S. Kolovos, Louis M. Rose, Nicholas Matragkas, Richard F. Paige, Esther Guerra, Jesús Sánchez Cuadrado, Juan De Lara, István Ráth, Dániel Varró, Massimo Tisi, and Jordi Cabot. 2013. A research roadmap towards achieving scalability in model driven engineering. In *Proceedings of the Workshop on Scalability in Model Driven Engineering* (BigMDE '13), Davide Di Ruscio, Dimitris S. Kolovos, and Nicholas Matragkas (Eds.). ACM, New York, NY, USA, , Article 2 , 10 pages. DOI=10.1145/2487766.2487768 http://doi.acm.org/10.1145/2487766.2487768