A Research Roadmap Towards Achieving Scalability in Model Driven Engineering

Dimitrios S. Kolovos[1], Louis M. Rose[1], Nicholas Matragkas[1], Richard F. Paige[1],
Esther Guerra[2], Jesús Sánchez Cuadrado[2], Juan De Lara[2],
István Ráth[3], Dániel Varró[3], Massimo Tisi[4], Jordi Cabot[4]
[1]University of York, [2]Universidad Autónoma de Madrid,
[3]Budapest University of Technology and Economics, [4]University of Nantes
{dimitris.kolovos, louis.rose, nicholas.matragkas, richard.paige}@york.ac.uk,
{Esther.Guerra, Jesus.Sanchez.Cuadrado, Juan.deLara}@uam.es,
{rath, varro}@mit.bme.hu, {massimo.tisi, jordi.cabot}@inria.fr

ABSTRACT

As Model-Driven Engineering (MDE) is increasingly applied to larger and more complex systems, the current generation of modelling and model management technologies are being pushed to their limits in terms of capacity and efficiency. Additional research and development is imperative in order to enable MDE to remain relevant with industrial practice and to continue delivering its widely recognised productivity, quality, and maintainability benefits. Achieving scalability in modelling and MDE involves being able to construct large models and domain-specific languages in a systematic manner, enabling teams of modellers to construct and refine large models in a collaborative manner, advancing the state of the art in model querying and transformations 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. This paper attempts to provide a research roadmap for these aspects of scalability in MDE and outline directions for work in this emerging research area.

1. INTRODUCTION

Modelling is an essential part of any engineering process. Engineers of all disciplines construct models of the systems they intend to build – e.g. software applications, bridges, airplanes – to capture, test, and validate their ideas with other stakeholders before embarking on a long and costly production process. Model-Driven Engineering (MDE) is a software engineering methodology that attempts to reduce the accidental complexity [1] of software systems by promoting *models* that focus on the *essential complexity* of systems, as first-class artefacts of the software development process. In contrast to traditional software development methodologies where models are mainly used for communication and post-mortem documentation purposes, in MDE models are the main living and evolving artefacts from which concrete software development artefacts can be produced in an automated fashion, through model-to-model and model-to-text transformation.

With object-oriented techniques having reached a *point of exhaustion* [2, 3], MDE constitutes the latest paradigm shift in software engineering as it raises the level of abstraction beyond that provided by 3rd generation programming languages. To avoid the pitfalls of one-size-fits-all CASE tools, MDE advocates domain-specific solutions and modern MDE architectures provide the means to implement these using a combination of general purpose languages (e.g. UML), domain specific languages, and languages and tools for automated model management (transformation, validation, comparison, merging, refactoring etc). In recent studies, MDE has been shown to increase productivity by as much as a factor of 10 [4, 5], and significantly enhance important aspects of the software development process such as maintainability, consistency and traceability [6].

As MDE is increasingly applied to larger and more complex systems, the current generation of modelling and model management technologies are being stressed to their limits in terms of their capacity to accommodate collaborative development, efficient management and persistence of models larger than a few hundreds of megabytes in size. As such, a new line of research is imperative in order to achieve scalability across the MDE technical space and enable MDE to remain relevant and continue delivering its widely-recognised productivity, quality, and maintainability benefits.

Scalability in software engineering has different dimensions: number of software engineers; size of engineering artefacts; size and complexity of languages used; size of engineering tasks that are carried out; number of engineering artefacts, etc. As illustrated in Figure 1, achieving scalability in MDE involves:

- being able to construct large models and domain specific languages in a systematic manner;
- enabling large teams of modellers to construct and refine large models in a collaborative manner;
- advancing the state of the art in model querying and transformations tools so that they can cope with large models (of the order millions of model elements);

BigMDE '13, June 17, 2013 Budapest, Hungary

- providing an infrastructure for efficient storage, indexing and retrieval of such models.

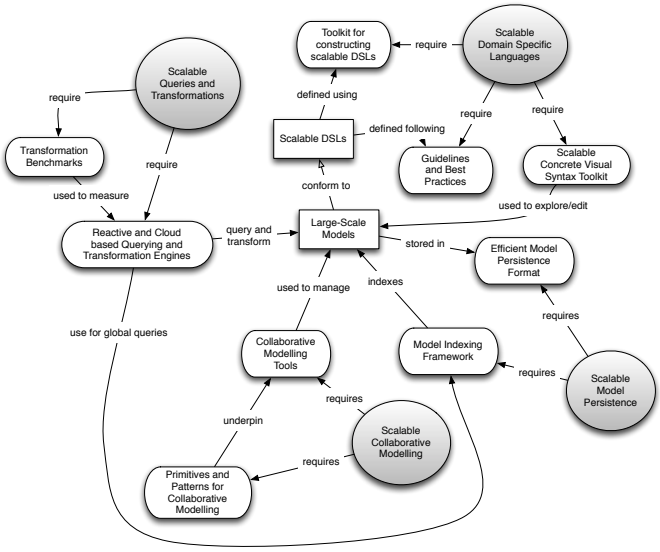

Figure 1: Tackling the challenge of scalability in MDE

The rest of the paper (Sections 2-5) provides an overview of the state of the art in these four key areas, identifies the main challenges that need to be overcome, and outlines directions for further research and development. Section 6 concludes the paper.

2. SCALABLE DOMAIN SPECIFIC LANGUAGES

Even though the concepts, techniques and tools for MDE have been notably improved over the last few years, we still find that models and languages do not scale well, heterogeneous languages (perhaps belonging to different technical spaces) are difficult to combine, and there are no satisfactory techniques for the application of MDE to large, complex systems in more complicated scenarios. These scenarios require techniques for the engineering, modularization and reuse of large models and complex modelling languages enabling their flexible combination. Many research groups have recognized these deficiencies [7, 8], making this area a very promising line of research, with great potential impact in industry. We review the current state of the art regarding scalable language design, and propose research directions to tackle the identified gaps.

2.1 State of the Art

There are several works aimed at defining compositional mechanisms for languages and models. The Reuseware project (`http://www.reuseware.org/`) aims at providing composition technology and techniques for languages lacking such built-in mechanisms. Reuseware is specifically aimed towards languages in the context of the Semantic Web, such as OWL, and modeling languages, like the UML. Such techniques allow for componentization and composition of artifacts written in these languages, fostering reuse [9]. In [10] the authors extend the OMG's Meta-Object Facility (MOF) for the specification of proper meta-model components with provided export and required import interfaces. This facilitates information hiding and enables the construction of languages by composing these components. However, the technique is only described theoretically and does not consider composition of models. In [11] the authors propose a similar approach for the definition of import and export interfaces for meta-models and models. Again, currently the approach is only defined theoretically, but an implementation atop EMF is foreseen. Moreover, none of these approaches tackle issues related to consistency checking when models are interconnected.

With respect to handling visualization of large, complex models, some researchers have brought abstraction techniques from the field of information visualization. For example, in [12], the authors develop an ad-hoc semantic-zooming technique to ease navigation in complex UML diagrams, and some visual language editors like DiaGen enable the definition of abstractions [13]. However, these abstractions have to be manually programmed for each different language. Some preliminary work in the definition of generic model abstractions has been reported in [14], which can be reused across different meta-models, however, there is still no support for visualizations, and these abstractions have not been applied in the context of large, heterogeneous models.

Finally, little work has addressed processes for developing and testing meta-models [15, 16]. For example, in [16] the authors propose a language to write automated tests for conceptual schemas, which could be used for test driven development of DSLs. In [15] the authors propose a process for the incremental development of meta-models by considering increasingly refined test models. Some other works (including proposals from some of the partners) have explored the induction of meta-models from example models [17, 18], and enabling collaboration in the definition of the meta-models [19]. However, none of these works consider issues related to scalability of models or meta-models.

2.2 Research Directions

The basic activities in Language and Model Engineering are the specification of domain-specific languages (normally through meta-models), the generation of modelling tools starting from those specifications, and the construction of models using the generated tools. Each of these techniques should be scalable for industrial use, and should enable the adequate handling of complexity. We believe that research is needed to facilitate the use of these techniques in contexts and applications of industrial scale. In particular, we foresee the following dimensions:

Scalable Language Design. In the first place, it should be possible to take into account scalability concerns when designing a modelling language. Appropriate techniques should be provided to extend the meta-model structure when large models are expected, in order to enhance efficiency for certain model queries. Such extensions could be suggested by an intelligent recommender system, on the basis of the expected sizes, queries and model usages. An additional mechanism is to include automated support for modularization, enabling the construction of libraries of reusable models and meta-model fragments (as it is done in the programming domain to tackle scalability). For example, including in the

meta-modelling infrastructure concepts like package, namespace, fragment, diagram or sub-model, which should be instantiated with controlled cardinality. Those model fragments, of potentially heterogeneous technical spaces, can be distributed and linked through soft references [20, 21, 22]. To enable the correct composition and reuse of fragments, (non-intrusive) mechanisms for the contract-based definition of template models and meta-models are needed. These decomposition and hierarchical constructs pose the problem of consistency and validation of global properties of the model. Hence efficient, incremental techniques for such validation are needed.

Scalable Concrete Syntaxes. Another means to tackle scalability is to provide support for useful model abstractions, providing a simplified view of a model, or introducing hierarchical elements, organizing models at different levels of abstraction. A means to facilitate their construction is to define generic abstractions, which can be reused across different modelling languages [14]. These techniques should be available both for the abstract syntax of models (i.e., "raw" models), and also for the graphical concrete syntax level, in order to facilitate the visualization and exploration of large-scale models, at different levels of abstraction. Another common issue is that no concrete syntax (only generic tree-based editors) is defined for some meta-models, which becomes problematic as models grow. To address such scenarios, research on automatic generation of graphical concrete syntaxes, adaptable to the characteristics of the meta-model, and taking scalability as a concern to enable usable navigation and visualization of large models is needed.

Heterogeneity. In real projects, one seldom uses isolated DSLs in the development, but several languages might be needed to describe the different aspects of the system to be built. These concerns could even be expressed using different technological spaces, like combinations of DSLs, UML or specialized languages like Matlab/Simulink. Hence, research on supporting the construction of multi-view language environments by reusing meta-model fragments, possibly of heterogeneous technologies, is needed.

Processes and Methodologies. Currently, DSLs and meta-models are often developed in an informal, ad-hoc way. However, being central to the MDE process, DSLs should be engineered using sound principles and methods, gathering requirements from all stakeholders. However, current MDE practice lacks proposals in this direction. Therefore, processes and methodologies enabling the engineering of DSLs, and the disciplined use of models are needed, using the techniques described above in an industrial setting.

3. SCALABLE QUERIES AND TRANSFORMATIONS

The proposed research agenda on accelerating model transformation engines builds on a vast literature of methods to improve the performance of computation systems.

3.1 State of the Art

Incrementality.

One of the fundamental techniques commonly used in this area is incrementality. The idea has already been applied to model transformations, and the most followed approach is offline incrementality. [23] proposes an automatic way to synchronize the source and target models of an ATL transformation offline. Incrementality is implemented by interfacing with existing differencing tools for calculating changes to the models and propagating them bidirectionally. Hearnden et al. [24] synchronize two models incrementally using a declarative logic-based transformation engine. The approach records a transformation execution and links each transformation step to the correspondent changes in the source model. This information is then used for change propagation. Live and offline incrementality has been already implemented with Graph Transformations techniques, for example in [25]. Especially the work in [26] implements live incrementality, based on the RETE algorithm, a well-known technique in the field of rule-based systems. These graph transformation approaches focus on incremental pattern-matching to improve the performances of the transformation. [27] employs Triple Graph Grammars for incremental offline model synchronization in both directions. With respect to model querying, the topic of evaluating OCL expressions incrementally has been investigated by Cabot [28], especially for detecting if a modification to a UML model violates OCL constraints that were satisfied before.

Lazy computation in model transformations.

While another well-known method to improve scalability is lazy computation, we are not aware of any model transformation tool with an on-demand generation strategy. The Stratego [29] system allows user-defined execution strategies for transformation rules. While user-defined strategies have been employed to implement target-driven approaches [30], the activation of rules as answer to external consumption has not been addressed. VIATRA [31], despite not implementing on-demand transformation, evaluates lazily the matches of connected rules to avoid unnecessary computation, as described in [32]. Outside the MDE domain, [33] provides an interpreter for XSLT that allows random access to the transformation result. They also show how their implementation enables efficient pipelining of XSLT transformations. The implementation of a lazy evaluator for functional (navigation) languages is a subject with a long tradition [34]. We refer to [35] for an example based on Lisp. This subject has been explored in [36] and in [37] where performance measures are presented.

Performance optimization for model transformations.

Other optimization techniques have been explored in model transformation engines [38, 39]. Lazy loading [40] is a complementary subject to lazy navigation, when dealing with models that do not fit into the memory available to the transformation engine. [41] explores the subsequent reuse of matches of transformations rules for performance tuning. In [42] model navigation results are cached to speed/up transformation sequences. Incremental transformations are often coupled with *retainment rules* that make it possible to avoid overriding manual changes performed to the target model when the correspondent part of the source model has been updated. Retainment rules for model transformations have been investigated in [43]. [41] presents methods to evaluate pattern matches of different rules in an overlapped way, to increase performance. In [42] transformation con-

text is preserved to efficiently perform incremental updates whereas in [38] and [39] strategies for the problem of graph pattern matching optimization are investigated.

3.2 Research Directions

To address these gaps it would be valuable to develop a novel transformation engine able to generate on demand the elements of models connected by a transformation network. We propose a shift of paradigm for programming model-driven applications towards reactive programming [44], and we perform the first steps in this direction by implementing and practically evaluating a reactive engine for model transformations. *Reactive programming* denotes a programming paradigm oriented to the propagation of changes through data flows. An example of reactive programming in this broad sense, is a language whose programs automatically update their computation whenever some input data changes. In our model-driven context we propose a paradigm where a network of *reactive transformations* defines persistent dataflows among models. A *reactive transformation engine* takes care of activating only the strictly needed computation in response to *updates* or *requests* of model elements. The reactive engine offers a combination of incremental and lazy computation (and loading) that transparently keeps the system consistent according to the provided transformation rules.

A reactive engine also opens the way to scenarios based on infinite intermediate models generated on demand, or streaming models propagating from inputs to outputs. This research has the potential to widen the application space of the model-driven approach to new scenarios.

4. SCALABLE COLLABORATIVE MODELLING

As model sizes and complexity have grown, multi-user environments (which are already standard in traditional, source-code centric development environments) are necessary. *Collaboration* and related features (such as version management, conflict resolution, model migration and merging) are now widely recognized as services of key importance, especially in terms of reusability and overall efficiency [45]. Unfortunately, the state of the art in modelling technologies has not yet caught up, as current tools leg behind both in terms of features and maturity. As a result, tool providers and toolchain designers have to rely on ad-hoc solutions. In the following, we provide a brief overview of these challenges.

4.1 State of the Art

Model repositories.

Model repositories are storage systems for modeling artefacts that are mostly focused on persistence and concurrent access over a distributed infrastructure (client-server). They have limited support for advanced use-cases such as conflict management, branching, model comparison etc. Instead, they provide extension mechanisms and core APIs that auxiliary, function-specific tools may use. In the following overview, we focus on the collaborative aspect of such technologies.

The Eclipse Modeling Framework Connected Data Objects (CDO [46]) framework is a model repository for EMF models primarily targeting scalable model persistence and version management, with a simple collaborative access layer. CDO implements a custom API for collaboration, based on transaction management and *views* that rely on a partitioning technique (implicit locking). Unfortunately, there is no mature support for conflict management and merging.

The enhancement of the collaboration features of CDO are the primary goal of the Dawn project [47] that should feature a collaborative UI and corresponding API. The aim is to provide preliminary collaboration primitives (such as locking, update, commit) for generated EMF and GMF editors.

MORSA [48, 49] is a recent approach for scalable model persistence based on a NoSQL back-end and on-demand loading/caching mechanisms, with a prototypical implementation for EMF models. Its primary focus is scalability and only provides preliminary query and integration facilities (does not cover access control, version management or security).

Online collaborative modelling systems.

Online collaborative modelling systems rely on a short transaction model, whereby a single, shared instance of the model is edited by multiple users in real time (i.e. all changes are propagated to all participants instantly). These systems lack conflict management, or only provide very light weight mechanisms (such as voluntary locking). As a result, conflicts are very limited in scope and are resolved instantly, at the cost of communication overhead and that all parties are required to be online simultaneously.

CoolModes (COllaborative Open Learning and MODEling System) [50] implements an online collaborative model editing approach for e-learning, with the prime focus on communication. It features a plugin architecture by which custom DSMLs can be specified using XML DTDs and integrated into the system.

EMFCollab [51] is an open-source effort that implements a light-weight and thus easy-to-integrate online collaboration feature over EMF models. The implementation is compatible with traditional file-based version control systems like CVS or SVN, as files are used for persistence.

The SpacEclipse-CGMF [52] approach is an attempt to integrate online collaborative functionality in Eclipse Graphical Modeling Framework-based graphical editors. It also integrates a model-based way to define both the domain of the graphical editor and the workspace configuration of the tool to be generated. This is done using a dedicated DSL.

Model versioning systems.

Model versioning systems are more closely aligned with offline version control systems (VCS) such as CVS or SVN. They follow the *long transaction model* whereby contributors are assumed to commit larger portions of work with respect to a certain (past) version as the reference. Hence, since conflicts are common, their detection, resolution and merging are features of top importance. Depending on their architecture, they may or may not deal with auxiliary services such as authentication and access control (these may be provided by additional middleware such as the web server that hosts the actual communication between server and client).

ModelCVS [53, 54] is one of the earliest attempts to create a CVS-like version control system for modelling artefacts. It is a collection of early prototype tool integration meta-tools (last updated in 2008), consisting of a model differencing

and matching framework, a model mapping editor and a semantic versioning back-end. The project did not produce technology of industrial maturity.

AMOR [55, 56] (formerly known as SmoVer) builds on the experiences gained in early projects like ModelCVS, with the focus on improvement of conflict management. AMOR provides precise conflict detection, intelligent conflict resolution within an adaptable versioning framework which versioning adapters of various DSMLs can extend as plugins. The detection and resolution of conflicts makes use of advanced techniques such as semantics or operation-driven reasoning [57], visualization, data mining and machine learning techniques. Unfortunately, AMOR has not evolved into a fully usable tool with only limited prototypes available; the final goal is to integrate the entire framework into the Enterprise Architect suite.

The Eclipse Modeling Team Framework (MTF) [58], an Eclipse project in the pre-proposal phase, is intended as the continuation of AMOR. MTF is intended as a meta-repository for modelling artefacts within the Eclipse workspace, leveraging existing Eclipse technologies such as the Team API, CDO and SVN integration to provide a fully integrated meta- and instance model versioning system.

EMFStore [59, 60] is an implemented and working model versioning framework for EMF that provides APIs for conflict management, but in a limited way: although some built-in prototypes are available, custom DSMLs need hand-coded and domain-specific solutions for reliable operation. EMFStore does not scale to large models and also uses an uses RDBMS backend through EMF Teneo, providing only very simple access control.

Model differencing.

In cases where the server-side VCS cannot be replaced by a custom solution for models, *offline model comparison, differencing and merging tools* such as EMF Compare 2.0 [61] or EMF Diff/Merge [62] are also often used (in combination with the traditional VCS). In these cases, the detection and resolution of conflicts is performed by a (dedicated) user manually on their workstation, by checking out all versions of the models, performing the comparison, resolving the conflicts (for which some hints may be provided by the diff-merge tools) and checking the result back in. Despite the sophistication of diff-merge algorithms especially in EMF Diff/Merge, a generic, domain-independent solution has not been developed yet, i.e. tool developers have to augment the core engine with their domain-specific customizations (just like in the case of EMFStore).

Convergence of approaches.

More recent developments in the (Eclipse) collaborative modelling community have resulted in convergence of approaches between model repositories, online collaboration engines and version control systems. CDO/Dawn, for instance, has been extended with online collaboration features and development is planned to converge the technological foundations of CDO, EMFStore, EMF Compare and EMF Diff/Merge.

In addition, more advanced, integrated frameworks such as ModelBus [63] have appeared to address multiple issues related to collaborative modelling (unfortunately, the development of ModelBus slowed down significantly in recent years, with version 2.0 still to appear as of late 2012, even though originally planned for 2009). ModelBus provides a tool integration layer for service-oriented tool orchestration [64] and integrates well-known industrial modelling tools (e.g. Rational Software Architect, Papyrus, Enterprise Architect), as well as auxiliary tools such as Rational DOORS, Microsoft Office and MatLab Simulink. Moreover, it also integrates prototype tools for model validation, metrics evaluation and traceability management. With respect to collaboration, ModelBus implements a basic model repository based on EMF with built-in model conversion to support non-EMF tools (these adapters have to be programmed manually). While the backend provides useful features such as notifications (e.g. model changes, but not model element changes), it only offers limited scalability (in terms of model size). ModelBus supports both offline and online collaboration through a very simple conflict management layer based on voluntary model element-level locking.

The key weaknesses of the collaborative modelling state of the art can be summarized as follows: (i) immature integration of online and offline collaboration patterns; (ii) mostly ad-hoc architectures that prohibit or make the implementation of domain-specific collaboration/version management difficult; (iii) very simplistic locking and conflict management solutions that severely hinder developer productivity; (iv) the lack of a flexible *and* scalable back-end platform that caters to both Eclipse-based and other (commercial) tools.

4.2 Research Directions

We foresee a multi-device collaborative modelling framework built on the model bus design pattern, as follows.

Support for online and offline collaboration. The framework should support both offline and online collaboration in a multi-user and multi-device environment, providing a model access layer (transaction management, queries, views and manipulation) featuring basic collaboration primitives (push, pull, commit, merge), and an adaptation layer for the integration of access control and security services.

Extensibility. It should be built on an extensible architecture that allows the integration of domain-specific, customized plugins for conflict management (detection, resolution and merging).

Locking and conflict management. As novel and innovative features, it should include:

- *query-driven dynamic locking* that uses complex graph queries [65] for the specification of locking partitions for views and manipulative transactions. Such queries should operate in a collaboration-aware manner that includes support for real-time updates and locked queries (where updates are propagated only from a pre-defined subset of collaboration partners).

- *automated conflict resolution* based on design-space exploration techniques [66] that are able to ensure domain consistency and well-formedness by automatically applying model manipulation policies to find valid and conflict-free model states.

Technology. The middleware should define a client-server protocol, core collaboration and version management operations (e.g. model manipulation, locking, branching, merging, upgrade) and extension mechanisms for various locking, conflict management and access control services. On the front-end, it should be compatible with existing and future Eclipse-based technologies (EMF and its auxiliaries and the Team API); on the back-end, it should fully support and integrate into the scalable model persistence framework.

5. SCALABLE MODEL PERSISTENCE

An essential component of scalable MDE is infrastructure that facilitates persistence and retrieval or large models in an efficient manner. This section reviews the state of the art in this area and the challenges that need to be overcome through further research and development.

5.1 State of the Art

The most widely adopted format for model persistence is the XML Metadata Interchange (XMI) format, which is an Object Management Group (OMG) and an ISO/IEC (19503:2005) standard. XMI was introduced in order to enhance interoperability between modelling tools and prevent vendor lock-in. Since its introduction, XMI has been adopted as a common import/export model persistence format by the majority of UML modelling tools (IBM RSA, Poseidon, MagicDraw UML, Modelio, Altova UModel etc.), and as a native persistence format in the Eclipse Modeling Framework (EMF) and the MetaData Repository (MDR). While XMI has been a significant step towards tool interoperability, as discussed in [67] and [8] it is not a particularly efficient model representation format, as – being based on top of XML – it provides limited support for lazy or partial model loading, features that are essential for managing large models.

To address the limitations of XMI with working with large models, several alternatives have been proposed. In [40], the Binary Model Syntax (BMS) is briefly discussed as a high performance binary alternative to XMI. However, since 2009, when the article above was published, there have not been any updates or releases of BMS in the public domain. The Connected Data Objects framework (CDO) [46] is a framework built on top of the Eclipse Modeling Framework, and supports persistence of large models in relational databases supporting features such as save points, explicit locking, change notification, queries, temporality, branching, merging, offline and fail-over modes. A major concern with CDO is that it implements its own version control management system and there are strong indications that this is hindering industrial adoption, as moving away from stable and proved version control management systems such as CVS, SVN and VSS and into a newly-developed VCS is not an easy decision. Also, recent work has demonstrated that CDO does not scale up as well as advertised. More specifically, CDO failed to load all test sets that were greater than 271MB in [67] although the documentation claims that it has been used to load models up to 4GB.

Other related work in the field of scalable model persistence, includes the Mongo-EMF [68] and the Morsa [67] systems which leverage NoSQL (Not Only SQL) database systems. Morsa achieves scalable model persistence by employing on-demand loading facilities that are able to retrieve and update model fragments on a per-need basis. Morsa has been demonstrated to out-perform XMI in terms of the memory needed to load and traverse a large model (646MB) by 17 times while requiring 20 times more time [67]. Mongo EMF is very similar to Morsa but no results have been reported on its performance yet. Both Mongo-EMF and Morsa are prototypes and so far there is no indication that they target or plan to target issues such as security and access control, which are critical for the deployment of such solutions in an industrial context.

5.2 Research Directions

Efficient Model Storage. The current standard model storage format is the XML Metadata Interchange[69]. As XMI is an XML-based format, in order to access any model elements using current state-of-the-art modelling frameworks such as EMF, the complete model file needs to be parsed and loaded in memory first. This implies that the larger the model file, the more time and memory is needed in order to load the model. Also, XMI inherits the verbosity of XML which means that XMI-encoded model files are much larger in size than needed in order to store the information they do.

To address these issues, we envision a new efficient model representation format that will reduce the size of model files, enable modelling and model management tools to lazily load the contents of a model into memory, and access specific model elements without needing to read the entire model file first. We anticipate that such a format will provide a dramatic improvement both in terms of both the size of model files, and in terms of the memory and time required to load these models. For instance, findings from the Google Protocol Buffers project suggest that a well-designed binary format can deliver improvements of a scale of 3-10 in terms of size, and 20-100 in terms of loading speed compared to XML. To design the proposed format, existing successful binary formats such as these provided by Google Protocol Buffers, BSON and Fast Infoset should be investigated in order to develop a solid understanding of their structure, strengths and weaknesses.

Model Indexing. With a faster and more efficient model persistence format that provides a reduction of the scale of 10 in terms of size, an XMI-based model of the order of hundreds of MBs, would now be of the order of tens of MBs. In a typical collaborative development environment where artefacts are stored in a central repository such as a Version Control System (VCS – e.g. CVS, SVN, Git etc.), an FTP server or a shared network folder, and synchronised over the network, even files of the order of tens of MBs are challenging to manage as for every change they need to be transferred back and forth between the local copy and the remote repository. Storing a large model as a single file can also be sub-optimal as it can cause frequent conflicts when using an optimistic locking VCS or lock-outs when using a pessimistic locking VCS.

Two solutions have been proposed for addressing this problem [8]:

1. Storing large models in dedicated model repositories that enable model-element level (instead of file-level) version control operations (check in, check out, lock etc.);

2. Splitting large models over many cross-referenced physical files (model fragments).

The first approach requires both a leap in terms of the modelling tools used to edit models, as the majority of modelling tools work with file-based models, and a transition from a robust and established types of repositories which work well with a wide range of development tools, to newly developed model-specific repositories. The particularly limited adoption of model-specific repositories such as CDO, and ModelCVS [53, 54] so far has demonstrated that industrial users can be reluctant to make such a drastic transition in practice. As such, and in order to provide industrially-relevant results, we will mainly focus on the second approach.

The main advantage of the second approach is that it works well with existing modelling tools (as the vast majority of them work with files), and with existing types of remote repositories (such as CVS, SVN, Git, FTP, shared network folders etc). However, using this approach with current state-of-the-art technologies makes it impossible to compute queries of global nature such as "find all classes that are sub-classes of X" without going through all the model fragments from the remote repository every time. To demonstrate this limitation, consider the scenario on the left side (a) of Figure 2. In this scenario, the VCS repository contains 3 model fragments (A, B and C) from which the developer has checked out only fragment A. Now, if the developer needs to know which other fragments in the repository reference its X element, they need to check out, load and examine every other fragment in the repository (B and C in this case). Obviously, as the number of model fragments in the repository grows, this approach becomes increasingly inefficient.

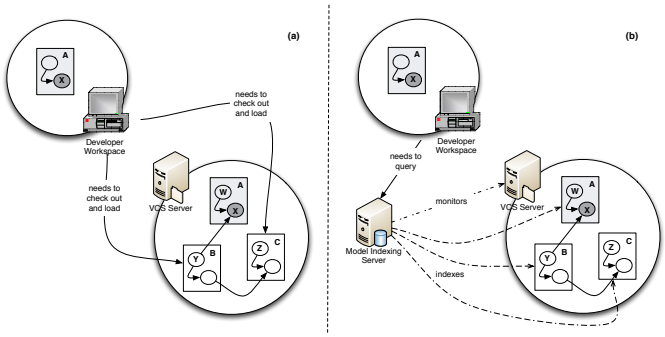

Figure 2: Performing global queries on model fragments stored in a VCS repository without (a) and with (b) an indexing server

To address this limitation, we envision a model indexing framework that can monitor the contents of remote version control repositories, and index the models they contain in scalable database that will enable efficient computation of global queries. The model indexing framework will support monitoring different types of remote repositories (SVN, CVS, Git, FTP, remote filesystems etc.) and indexing of heterogeneous models (i.e. XMI, binary, text-based models) using a driver-based architecture.

Applications of such a framework can extend beyond the boundaries of MDE as it can also be used in order to index other types of artefacts, including source code (e.g. Java source files can be loaded as models conforming to the Java metamodel using tools such as Jamopp[1]). Indexing the source code of an entire repository (and even of multiple repositories) would enable developers and tools to perform global queries (e.g. which classes inherit from class X? if I change the signature of method Y, which classes will be affected? is there any class named Z or that has a method Q?) without needing to check out locally all the code from these repositories – which is currently not possible with existing technologies.

In order for the framework to also apply to the source code indexing problem discussed above, the persistence mechanism that will be used to underpin the index needs to be highly scalable. While this is still an open research question, NoSQL solutions such as Neo4J, OrientDB, Cassandra and MongoDB appear to be promising candidates. The language that clients (e.g. modelling or model management tools) will use in order to query the model indexing framework is also an open research question.

Security and Access Control. In terms of security, the indexing framework needs to leverage the authentication mechanisms provided by the remote repository it indexes, and ensure that access control rules in the index are consistent with those of the repository. For instance, in the example of Figure 2, if the user that performs the query is not permitted to access model fragment B in the VCS, the query should either not return B or inform the user that X is referenced from another model that they don't have access to. The precise definition of the security policy is again an open research question that requires further investigation.

6. CONCLUSIONS

In this paper we have identified a number of challenges related to scalability in Model Driven Engineering, we have discussed the state of the art in the areas of scalable language development, model querying and transformation, collaborative modelling and persistence and we have proposed directions for further research in this area which we plan to explore further in the future.

Acknowledgements

We would like to thank Scott Hansen (The Open Group), Alessandra Bagnato (SOFTEAM), Pedro Maló (Uninova), Vincent Hanniet (Soft-Maint) and Salvator Trujillo (IKERLAN) for their help with identifying the challenges related to scalable MDE from an industrial perspective, and for their contributions to shaping the proposed research directions.

[1]http://www.jamopp.org/

7. REFERENCES

[1] Frederick P. Brooks, Jr. No silver bullet essence and accidents of software engineering. *Computer*, 20(4):10–19, April 1987.

[2] Jean Bezivin. On the Unification Power of Models. *Software and System Modeling (SoSym)*, 4(2):171–188, 2005.

[3] D. Schmidt. Model-driven engineering. *IEEE Computer*, 39(2):25, 2006.

[4] Jaaksi, A. Developing Mobile Browsers in a Product Line. *IEEE Software*, pages 73–80, July/August 2002.

[5] J Karna, J-P Tolvanen, S Kelly. Evaluating the Use of Domain-Specific Modeling in Practice. In *Proc 9th Workshop on Domain-Specific Modeling*, 2009.

[6] Parastoo Mohagheghi, Vegard Dehlen. Where is the Proof? - A Review of Experiences from Applying MDE in Industry. In Schieferdecker, Ina and Hartman, Alan, editor, *Proc. 4th European Conference on Model Driven Architecture, Foundations and Applications (ECMDA-FA)*, volume 5095 of *Lecture Notes in Computer Science*, pages 432–443. Springer Berlin / Heidelberg, 2008.

[7] Mikaël Barbero, Frédéric Jouault, and Jean Bézivin. Model driven management of complex systems: Implementing the macroscopes vision. In *15th Annual IEEE International Conference and Workshop on Engineering of Computer Based Systems (ECBS 2008)*, pages 277–286. IEEE Computer Society, 2008.

[8] Dimitris Kolovos and Richard Paige and Fiona Polack. The Grand Challenge of Scalability for Model Driven Engineering. In Chaudron, Michel, editor, *Models in Software Engineering*, volume 5421 of *Lecture Notes in Computer Science*, pages 48–53. Springer Berlin / Heidelberg, 2009.

[9] Florian Heidenreich, Jakob Henriksson, Jendrik Johannes, and Steffen Zschaler. On language-independent model modularisation. *T. Aspect-Oriented Software Development VI*, 6:39–82, 2009.

[10] Ingo Weisemöller and Andy Schürr. Formal definition of mof 2.0 metamodel components and composition. In *Model Driven Engineering Languages and Systems, 11th International Conference*, volume 5301 of *Lecture Notes in Computer Science*, pages 386–400. Springer, 2008.

[11] Stefan Jurack and Gabriele Taentzer. A component concept for typed graphs with inheritance and containment structures. In *Graph Transformations - 5th International Conference, ICGT 2010*, volume 6372 of *Lecture Notes in Computer Science*, pages 187–202. Springer, 2010.

[12] Mathias Frisch, Raimund Dachselt, and Tobias Brückmann. Towards seamless semantic zooming techniques for UML diagrams. In *SOFTVIS*, pages 207–208. ACM, 2008.

[13] Oliver Köth and Mark Minas. Structure, abstraction, and direct manipulation in diagram editors. In *Diagrams*, volume 2317 of *LNCS*, pages 290–304. Springer, 2002.

[14] Juan de Lara, Esther Guerra, and Jesús Sánchez Cuadrado. Abstracting modelling languages: A reutilization approach. In *Proc. CAiSE 2012*, volume 7328 of *Lecture Notes in Computer Science*, pages 127–143. Springer, 2012.

[15] Antonio Cicchetti, Davide Di Ruscio, Dimitris Kolovos, and Alfonso Pierantonio. A test-driven approach for metamodel development. In *Emerging Technologies for the Evolution and Maintenance of Software Models*, pages 319–342. IGI Global, 2012.

[16] Albert Tort and Antoni Olivé. An approach to testing conceptual schemas. *Data Knowl. Eng.*, 69(6):598–618, 2010.

[17] Hyun Cho, Yu Sun, Jeff Gray, and Jules White. Key challenges for modeling language creation by demonstration. In *ICSE'11 Workshop on Flexible Modeling Tools*, 2011.

[18] Jesús Sánchez Cuadrado, Juan de Lara, and Esther Guerra. Bottom-up meta-modelling: An interactive approach. In *Model Driven Engineering Languages and Systems - 15th International Conference, MODELS 2012, Innsbruck, Austria, September 30-October 5, 2012. Proceedings*, volume 7590 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2012.

[19] Javier Luis Cánovas Izquierdo and Jordi Cabot. Community-driven language development. In *Proc. MISE'12 at ICSE*, 2012.

[20] Louis Rose, Dimitrios Kolovos, Nicholas Drivalos, James Williams, Richard Paige, Fiona Polack, and Kiran Fernandes. Concordance: A framework for managing model integrity. In *Modelling Foundations and Applications*, volume 6138 of *LNCS*, pages 245–260. Springer Berlin / Heidelberg, 2010.

[21] Ábel Hegedüs, Ákos Horváth, István Ráth, and Dániel Varró. Query-driven soft interconnection of emf models. In *Model Driven Engineering Languages and Systems - 15th International Conference, MODELS 2012, Innsbruck, Austria. Proceedings*, volume 7590 of *LNCS*, pages 134–150, 2012.

[22] Cauê Clasen, Frédéric Jouault, and Jordi Cabot. Virtual Composition of EMF Models. In *7èmes Journées sur l'Ingénierie Dirigée par les Modèles (IDM 2011)*, 2011.

[23] Yingfei Xiong, Dongxi Liu, Zhenjiang Hu, Haiyan Zhao, Masato Takeichi, and Hong Mei. Towards automatic model synchronization from model transformations. *Proc. of ASE'07*, page 164, 2007.

[24] D. Hearnden, M. Lawley, and K. Raymond. Incremental model transformation for the evolution of model-driven systems. *LNCS*, 4199:321, 2006.

[25] G. Bergmann, I. Ráth, and D. Varró. Parallelization of graph transformation based on incremental pattern matching. *Electronic Communications of EASST*, 18, 2009.

[26] Gabor Bergmann, Istvan Rath, Gergely Varro, and Daniel Varro. Change-driven model transformations. *Software and Systems Modeling*, pages 1–31, 2011. 10.1007/s10270-011-0197-9.

[27] Holger Giese and Robert Wagner. From model transformation to incremental bidirectional model synchronization. *Software & Systems Modeling*, 8(1):21–43, 2008.

[28] J. Cabot and E. Teniente. Incremental evaluation of OCL constraints. *Lecture Notes in Computer Science*, 4001:81, 2006.

[29] Eelco Visser. Program transformation with Stratego/XT: Rules, strategies, tools, and systems in Stratego/XT 0.9. In *Domain-Specific Program Generation*, volume 3016 of *LNCS*, pages 216–238. Springer, 2003.

[30] Jonne Van Wijngaarden and Eelco Visser. Program transformation mechanics: A classification of mechanisms for program transformation with a survey of existing transformation systems. Technical report, UU-CS, 2003.

[31] Dániel Varró and András Balogh. The model transformation language of the VIATRA2 framework. *Science of Computer Programming*, 68(3):214–234, October 2007.

[32] Gabriele Taentzer, Karsten Ehrig, Esther Guerra, J. de Lara, L. Lengyel, Tihamer Levendovszky, Ulrike Prange, D. Varró, and S. Varró-Gyapay. Model transformation by graph transformation: A comparative study. In *Proc. Workshop Model Transformation in Practice*, 2005.

[33] Steffen Schott and Markus L. Noga. Lazy XSL transformations. In *ACM Symposium on Document Engineering*, pages 9–18. ACM, 2003.

[34] Paul Hudak, John Hughes, Simon L. Peyton Jones, and Philip Wadler. A history of Haskell: being lazy with class. In *HOPL*, pages 1–55. ACM, 2007.

[35] Peter Henderson and James H. Morris, Jr. A lazy evaluator. In *Proceedings of the 3rd ACM SIGACT-SIGPLAN symposium on Principles on programming languages*, POPL '76, pages 95–103. ACM, 1976.

[36] Olivier Beaudoux, Arnaud Blouin, Olivier Barais, and Jean-Marc Jézéquel. Active operations on collections. In *MoDELS*, volume 6394 of *LNCS*, pages 91–105. Springer, 2010.

[37] Manuel Clavel, Marina Egea, and Miguel Angel García de Dios. Building an efficient component for OCL evaluation. *ECEASST*, 15, 2008.

[38] Gergely Varró, Katalin Friedl, and Dániel Varró. Adaptive graph pattern matching for model transformations using model-sensitive search plans. *Electr. Notes Theor. Comput. Sci.*, 152:191–205, 2006.

[39] Rubino Geiß, Gernot Veit Batz, Daniel Grund, Sebastian Hack, and Adam Szalkowski. GrGen: A fast SPO-based graph rewriting tool. In *ICGT*, volume 4178 of *LNCS*, pages 383–397. Springer, 2006.

[40] Jouault, Frédéric and Bézivin, Jean and Barbero, Mikaël. Towards an advanced model-driven engineering toolbox. *Innovations in Systems and Software Engineering*, 5:5–12, 2009.

[41] Tamás Mészáros, Gergely Mezei, Tihamer Levendovszky, and Márk Asztalos. Manual and automated performance optimization of model transformation systems. *STTT*, 12:231–243, 2010.

[42] David Hearnden, Michael Lawley, and Kerry Raymond. Incremental model transformation for the evolution of model-driven systems. In *MoDELS*, volume 4199 of *LNCS*, pages 321–335. Springer, 2006.

[43] T. Goldschmidt and A. Uhl. Retainment Rules for Model Transformations. In *1st International Workshop on Model Co-Evolution and Consistency Management at Models 2008*, 2008.

[44] D. Harel and A. Pnueli. *On the development of reactive systems*, pages 477–498. Springer-Verlag New York, Inc., New York, NY, USA, 1985.

[45] Kerstin Altmanninger, Martina Seidl, and Manuel Wimmer. A survey on model versioning approaches. *International Journal of Web Information Systems (IJWIS)*, 5(3):271–304, 2009.

[46] Eclipse. The connected data objects model repository (CDO) project, 2012. `http://eclipse.org/cdo`.

[47] Martin Fluegge et al. Dawn: Collaborative modeling with CDO, 2011. `http://wiki.eclipse.org/Dawn`.

[48] Javier Espinazo Pagan, Jesus Sanchez Cuadrado, and Jesus García Molina. Morsa: A scalable approach for persisting and accessing large models. In Jon Whittle, Tony Clark, and Thomas Kï£¡hne, editors, *Model Driven Engineering Languages and Systems*, volume 6981 of *Lecture Notes in Computer Science*, pages 77–92. Springer Berlin / Heidelberg, 2011. 10.1007/978-3-642-24485-8_7.

[49] Javier et al. Espinazo Pagán. Morsa: a NoSQL-based model persistence solution, 2012. `http://modelum.es/trac/morsa/`.

[50] Niels Pinkwart. A Plug-In Architecture for Graph Based Collaborative Modeling Systems. In V. Aleven et al, editor, *Supplementary Proceedings of the 11th Conference on Artificial Intelligence in Education, Sydney (Australia)*, pages 89–94, Sydney, Australia, 2003. SIT.

[51] Andras Schmidt et al. Emfcollab, 2011. `http://qgears.com/products/emfcollab`.

[52] Jesus Gallardo, Ana I. Molina, Crescencio Bravo, Miguel A. Redondo, and Cesar A. Collazos. An ontological conceptualization approach for awareness in domain-independent collaborative modeling systems: Application to a model-driven development method. *Expert Systems with Applications*, 38(2):1099 – 1118, 2011. Intelligent Collaboration and Design.

[53] G. Kramler, G. Kappel, T. Reiter, E. Kapsammer, W. Retschitzegger, and W. Schwinger. Towards a semantic infrastructure supporting model-based tool integration. In *Proceedings of the 2006 international workshop on Global integrated model management*, GaMMa '06, pages 43–46, New York, NY, USA, 2006. ACM.

[54] The ModelCVS project. A semantic infrastructure for model-based tool integration, 2006. `http://modelcvs.org`.

[55] Kerstin Altmanninger, Gerti Kappel, Angelika Kusel, Werner Retschitzegger, Martina Seidl, Wieland Schwinger, and Manuel Wimmer. Amor - towards adaptable model versioning. In *1st Int. Workshop on Model Co-Evolution and Consistency Management, in conjunction with Models'08*, 2008. Vortrag: 1st International Workshop on Model Co-Evolution and Consistency Management, Toulouse, Frankreich; 2008-09-30.

[56] The AMOR project. Adaptible model versioning project website, 2009. `http://modelversioning.org`.

[57] Inc. Pentaho. Conflict resolution reasoner, 2010. `http://pentaho.com`.

[58] Eclipse. Modeling team framework proposal, 2011. `http://www.eclipse.org/proposals/mtf/`.

[59] Maximilian Koegel and Jonas Helming. Emfstore: a model repository for emf models. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2*, ICSE '10, pages 307–308, New York, NY, USA, 2010. ACM.

[60] Eclipse. Emfstore project, 2011. `http://eclipse.org/emfstore`.

[61] C. Brun et al. EMF compare 2.0, 2012. `http://www.eclipse.org/emf/compare/`.

[62] O. Constant. EMF Diff/Merge, 2012. `http://eclipse.org/diffmerge/`.

[63] T. Ritter et al. ModelBus: a model-driven tool integration framework to build a seamlessly integrated tool environment for system engineering processes., 2012. `http://www.modelbus.org/modelbus/`.

[64] Eric Armengaud, Markus Zoier, Andreas Baumgart, Matthias Biehl, DeJiu Chen, Gerhard Griessnig, Christian Hein, Tom Ritter, and Ramin Tavakoli Kolagari. Model-based toolchain for the efficient development of safety-relevant automotive embedded systems. Technical Report 2011-01-0056, SAE International, Warrendale, PA, April 2011.

[65] Gábor Bergmann, Ákos Horváth, Istvan Rath, and Dániel Varró. Incremental evaluation of model queries over EMF models. In Dorina Petriu, Nicolas Rouquette, and Oystein 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.

[66] Ábel Hegedüs, Ákos Horváth, István Ráth, and Dániel Varró. A model-driven framework for guided design space exploration. In *26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, Lawrence, Kansas, USA, 11/2011 2011. IEEE Computer Society, IEEE Computer Society. ACM Distinguished Paper Award, Acceptance rate: 15%.

[67] Pagán, Javier Espinazo and Cuadrado, Jesús Sánchez and Molina, Jesús García. Morsa: a scalable approach for persisting and accessing large models. In *Proceedings of the 14th international conference on Model driven engineering languages and systems*, MODELS'11, pages 77–92, Wellington, New Zealand, 2011. Springer-Verlag.

[68] Bryan Hunt. Mongo-EMF, 2011. https://github.com/BryanHunt/mongo-emf/.

[69] Object Management Group. XML Metadata Interchange 2.0 Specification. http://www.omg.org/technology/documents/formal/xmi.htm.