Advanced Techniques and Tools for Secure Collaborative Modeling

Csaba Debreceni
Budapest University of Technologies and Economics, Department of Measurement and Information Systems
MTA-BME Lendület Research Group on Cyber-Physical Systems
Email: {debreceni}@mit.bme.hu

***Abstract*—Model-based systems engineering of critical cyber-physical systems necessitates effective collaboration between different collaborators, teams, stakeholders. Engineering artifacts stored in model repositories are concurrently developed in either offline (checkout-modify-commit) or online (GoogleDoc-style) scenario where the confidentiality and integrity of design artifacts need to be protected by access control policies. Unfortunately, traditional approaches for managing concurrent code development do not naturally extend to collaborative modeling which implies novel challenges.**

My research focuses on developing (i) a general secure collaboration scheme that guarantees that high-level access control policies are respected during collaboration and it can be integrated into existing version control systems (e.g. SVN) to support offline scenario; (ii) automated merging and fine-grained locking to enhance the efficiency of conflict resolution and prevention upon concurrent modification of the models; (iii) derivation and incremental maintenance of view models to provide specific focus of the designers by abstracting from unnecessary details of the underlying system model.

I. Problem and Motivation

The adoption of model driven engineering (MDE) by system integrators (like airframers or car manufacturers) has been steadily increasing in the recent years [1], since it enables to detect design flaws early and generate various artifacts (source code, documentation, configuration tables, etc.) automatically from high-quality system models.

The use of models also strengthens collaboration between distributed teams of different stakeholders (system integrators, software engineers of component providers/suppliers, hardware engineers, certification authorities, etc.) via model repositories, which significantly enhances productivity and reduces time to market. An emerging industrial practice of system integrators is to outsource the development of various design artifacts to subcontractors in an architecture-driven supply chain.

Collaboration scenarios include traditional *offline collaborations* with asynchronous long transactions (i.e. to check out an artifact from a version control system and commit local changes afterwards) as well as *online collaborations* with short and synchronous transactions (e.g. when a group of collaborators simultaneously edit a model). Even though, various collaborative modeling frameworks (like [2], [3], etc.) exist to support such scenarios, additional challenges arise that cannot be naturally extended from traditional code-based approaches due to the graph-like nature of the artifacts.

This paper is partially supported by the EU Commission with project MONDO (FP7-ICT-2013-10), no. 611125. and the MTA-BME Lendület 2015 Research Group on Cyber-Physical Systems.

A. Secure Collaborative Modeling

An increased level of collaboration in a model-driven development process introduces additional confidentiality challenges to sufficiently protect the intellectual property of the collaborating parties, which are either overlooked or significantly underestimated by existing initiatives. Even within a single company, there are teams with differentiated responsibilities, areas of competence and clearances described by high-level access control policies. Such processes likewise demand confidentiality and integrity of certain modeling artifacts.

Existing practices for managing access control of models rely primarily upon the access control features of the back-end repository. *Coarse-grained access control policies* aim to restrict access to the files that store models. For instance, EMF models can be persisted as standard XMI documents, which can be stored in repositories providing file-based access and change management (as in SVN [4], CVS [5]). *Fine-grained access control policies*, on the other hand, may restrict access to the model on the row level (as in relational databases) or triple level (as in RDF repositories). Unfortunately, coarse-grained security policies are captured directly on the storage (file) level often result in inflexible fragmentation of models in collaborative scenarios.

As a result, coarse-grained access control can lead to significant model fragmentation, which greatly increases the complexity of storage and access control management. In industrial practice, automotive models may be split into more than 1000 fragments, which poses a significant challenge for tool developers. Some model persistence technologies (such as EMFs default XMI serialization) do not allow model fragments to cyclically refer to each other, putting a stricter limit to fragmentation. Hence, MDE use cases often demand the ability to define access for each object (or even each property of each object) independently.

Furthermore, coarse-grained access control lacks flexibility, especially when accessing models from heterogeneous information sources in different collaboration scenarios. For instance, they disallow type-specific access control, i.e., to grant or restrict access to model elements of a specific type

(e.g., to all classes in a UML model), which are stored in multiple files.

My first research question is constructed as follows:

RQ-1 How to capture and enforce high-level access control policies during collaborative modeling?

B. Conflict Prevention and Resolution

Enabling a high degree of concurrent edits for collaborators is required to make the traditionally rigid development processes more agile. The increasing number of collaborators concurrently developing artifacts increases the probability of introducing conflicts. *Conflict avoidance* techniques such as locks try to prevent conflicts by letting the users request that certain engineering artifacts should be made unmodified by all other participants for a duration of time. But it usually leads to unnecessary preventions (locks) which significantly limits the degree of concurrent development and does not scale with the increasing number of collaborating teams. *Model merging* aims to resolve the conflicts, but, it can be complex tasks as the interdependence within a model makes conflicts easy to introduce and hard to resolve. Furthermore, domain-specific conflict resolution strategies are rarely taken into consideration in industrial frameworks (e.g. EMF Compare[6], EMF Diff/Merge[7]), hence the well-formedness of merge results is questionable.

My second research question is the following.

RQ-2 How to provide fine-grained prevention and automatized resolution strategies of conflicts?

C. Bidirectional Synchronization of View Models

Views are key concepts of domain-specific modeling in order to provide task-specific focus (e.g., power or communication architecture of a system) to engineers by creating a model which highlights only some relevant aspects of the system to help detect conceptual flaws. Typically multiple view models are defined for a given an underlying source model, which need to be refreshed automatically (or upon user request) upon changes in the source model.

Usually, these views are represented as models themselves (view models), computed from the source model. On one hand, the efficient forward propagation of changes from the source model to the views is challenging, as recalculating the view from scratch has to be avoided to achieve scalability. On the other hand, the efficient backward propagation of complex changes from one or more abstract view models to the underlying source model resulting in valid and well-formed models is also a challenging task which requires to limit the propagation to a well-defined part of the source model to achieve scalability.

My third research question is as follows.

RQ-3 How to derive and incrementally maintain view models and trace back complex changes to the underlying source models?

II. Preliminaries

A. Related Work

1) Secure Collaborative Modeling: Traditional version control systems (like [4]) adopt file-level access policies, which are clearly insufficient for fine-grained access control specifications. [2] allows for role-based access control with type-specific (class, package and resource-level) permissions, but disallows instance level access control policy specifications. Access control is not considered in recent collaborative modeling environments like [8], [9], [10], [3], [11], [12], or the tools developed according to [13]. [14] provides fine-grained role-based access control for online collaboration but no offline scenario is supported, though. Both online collaboration and role-based access control with type-specific (class, package and resource-level) permissions is provided in [2], but no facility for instance level access control policy specifications. However, there is a pluggable access control mechanism that can specify access on the object level.

2) Locking Support: The state-of-the-art locking techniques are the fragment-based and object-based locks. *Fragment-based locking* requires that models are partitioned into storage fragments, e.g. files or projects and entire fragments can be locked at once. *Object-based locking* locks individual model objects (including their attributes and connections) which requires to inspect the structure of the model.

Existing collaborative modeling tools either lack locking support or implement rigid strategies such as fragment-based locking, or locking subtrees or elements of a specific type, which hinder effective collaboration. Most of *offline collaborative modeling tools* [5], [15], [3], rely on traditional version control systems using file-based (same as fragment-based) locking with contributors committing large deltas of work. *Model repositories* [2], [9], support both implicit and explicit locking of subtrees and sets of elements. These locks can prevent others from modifying elements to avoid conflicts. *Online collaborative modelling frameworks* [11], [8], [10], [14], rely on a short transaction model: a single, shared instance of the model is concurrently edited by multiple users, with all changes propagated to all participants instantaneously. These approaches use timestamped operations to resolve conflicts or provide only lightweight lock mechanisms, e.g., explicit locks to certain elements.

3) Conflict Resolution in Model Artifacts: Model comparison refers to identifying the differences between models. Based on its result, *model merge* synthesizes a combined model which reconciles the identified differences. My research focuses on *three-way merge*, which uses the common ancestor O of local copy L and remote copy R to derive the merged model M. To determine the changes executed on O, a comparison is conducted between $O \leftrightarrow L$ and $O \leftrightarrow R$. The solution of merge M is obtained by applying a combination of changes performed either on L or R to the original model O.

Most approaches [6], [7], [16], [17], [18] are semi-automated as they use a two-phase process: (i) first, they apply the non-conflicting operations and then (ii) let the user

prioritize and select the operation to apply in case of two conflicting changes. This always results in a single solution due to the manual resolution by the user. In comparison, [19], [20] resolve the conflicts automatically in different ways and offer several solutions.

4) Incremental Maintenance of View Models.: View maintenance by incremental and live QVT transformations is used in [21] to define views from runtime models. The proposed algorithm operates in two phase, starting in check-only mode before an enforcement run, but its scalability is demonstrated only on models up to 1000 elements. [8] allows the composition of multiple EMF models into a virtual model based on a composition metamodel, and provides both a model virtualization API and a linking API to manage these models. The approach is also able to add virtual links based on composition rules. In [22], an ATL-based method is presented for automatically synchronizing source and target models. In [23], correspondences between models are handled by matching rules defined in the Epsilon Comparison Language, but incremental derivation is not discussed.

5) Backward Propagation: For the backward propagation of changes, the use of traceability links is a well-accepted approach to define which part of the source model has to be updated upon a change on the target model. In [24], these links are stored as a *correspondence model* where their maintenance is derived from the TGG rules. [25] also specifies *trace* classes to facilitate and maintain traceability links. [26] stores traceability links in Alloy[27] as a bijective mapping. [28] uses a *weaving* model that stores the traces of references between different models in the view, however all objects in the view model act as proxies to an object in the source model.

B. Foundational Techniques

1) Graph Patterns: A *graph pattern* represents structural constraints prescribing the interconnection between nodes and edges of given types extended with algebraic expressions to define attribute constraints. *Pattern parameters* are a subset of nodes and attributes representing the model elements interesting from the perspective of the pattern user. A *match* of a pattern is a tuple of pattern parameters that has the same structure as the pattern and satisfies all structural and attribute constraints.

2) Design Space Exploration: *Design space exploration (DSE)* aims to find optimal design candidates of a domain with respect to different objectives where design candidates are constrained by complex structural and numerical restrictions (e.g. described by *graph pattern*) and are reachable from an initial model by applying a sequence of exploration rules.

III. Overview of the Approaches

A. General Secure Collaboration Scheme

Approach. In [29], we proposed a query-based approach for modeling fine-grained access control policies, and we defined bidirectional model transformations to (i) derive filtered views (*front models*) for each collaborator from the original model

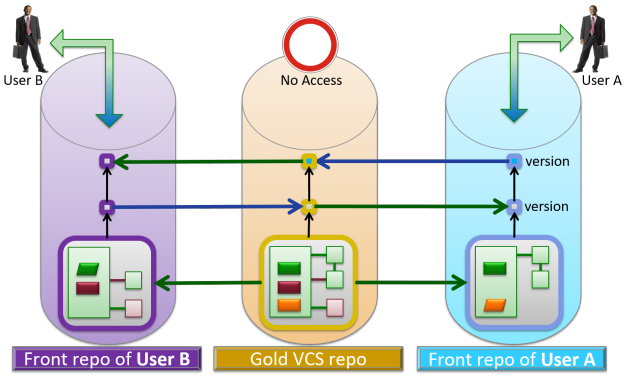

Fig. 1. MONDO Offline Collaboration - Architecture

(*gold model*) containing all the information and to (ii) propagate changes introduced into these views back to a server in both *online* and *offline* scenarios. Access control policies consist of rules that allow, obfuscate or deny read and/or write permissions of model parts identified by graph patterns detailed in [30].

In [31], a collaboration scheme between the clients of multiple collaborators and exactly one server is described to support fine-grained access control in offline scenario. The server stores the gold models and the clients can download their specific front models. Modifications, executed by a clients, are submitted to the server and they are accepted if write permissions are successfully checked. Right after the submission, the changes are propagated to the other front model while read permissions are enforced. Finally, clients can downloaded their updated front models.

The scheme is realized by extending *SVN*[4] using its hooks. The server and clients are realized as a *gold repository* and multiple *front repositories*, respecively. The *gold repository* contains *gold models*, but it is not accessible to collaborators. Each collaborator is assigned to a specific *front repository* containing a full version history of the front models. Change propagations are maintained between the repositories. As a result, each collaborator continues to work with a dedicated VCS as before, thus they are unaware that this front repository may contain filtered and obfuscated information.

My contributions related to the fulfillment of *RQ-1* :

Contribution 1 I proposed a generic modeling language to capture fine-grained access control policies integrated into a provenly secure collaborative architecture.

C1.1 *Access Control Language.* I proposed a rule-based access control language to describe high-level and fine-grained policies in both online and offline scenarios. Rules may allow, obfuscate or deny read and/or write permissions of model parts identified by graph patterns[30], [31].

C1.2 *Read and Write Dependencies.* I analyzed read and write dependencies implied by high-level access control policies as read and write permissions of a model part

may depend on other model parts implied by internal consistency rules [30].

C1.3 *Formalization of Transformation Rules.* I formalized transformation rules to derive secure front models with respect to the read and write permissions [31].

C1.4 *Secure Collaboration Scheme.* I formalized a collaboration scheme as *communicating sequential processes* (CSP) to enforce high-level access control policies. I specified correctness criteria and proved the correctness of the scheme [31].

C1.5 *Realization of Secure Collaboration.* I realized the collaboration scheme in case of offline scenarios by extending an existing version control system to enforce fine-grained access control while collaborators can use off-the-shelf tools [32], [31].

C1.6 *Evaluation.* I evaluated the scalability of the collaboration architecture on a case study of offshore wind turbine controllers [29], [32], [33], [31].

The bidirectional transformation and the algorithm to derive effective permission based on the proposed language is the contribution of Gábor Bergmann whereas the concept of the common architecture to support both online and offline scenarios is the contribution of István Ráth.

Uniqueness. Our provenly correct collaboration scheme is able to enforce fine-grained access control policies of modeling artifacts over existing version control system in case of offline scenarios. The scheme and its realization is demonstrated in [32] as an integration with SVN[4].

B. Conflict Reduction and Handling

Approach. In our preliminary work [34], we introduced the concept of property-based locking where collaborators request locks specified as a property of the model which need to be maintained as long as the lock is active. Hence, other collaborators are permitted to carry out any modifications that do not violate the defined property of the lock. In [35], the realization of property-based locking strategy is proposed as a common generalization of existing fragment-based and object-based locking approaches. Complex properties are described as graph patterns to express structural (and attribute) constraints for a model where the result set, i.e. the matches of graph pattern, can be calculated by pattern matchers or query engines. Only those modifications are allowed that do not change the result set of a list of queries as depicted in Fig. 2.

In [36], we proposed DSE-Merge that exploits guided rule-based *design space exploration (DSE)* [37] to automate the three-way model merge with an architecture depicted in Fig. 3. Three-way model merge is applied to DSE problem where the *initial model* consists of the original model O and two difference models (ΔL and ΔR); the *goal* is that there are no executable changes left in ΔL and ΔR; *operations* are defined by change driven transformation rules to process generic composite (domain-specific) operators; and *constraints* may identify inconsistencies and conflicts to eliminate certain trajectories. The output is a *set of solutions* consisting of

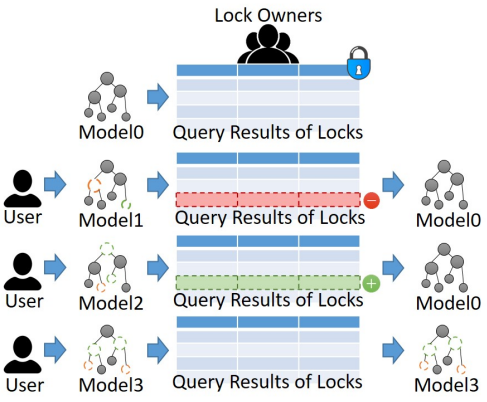

Fig. 2. Behavior of Property-Based Locks

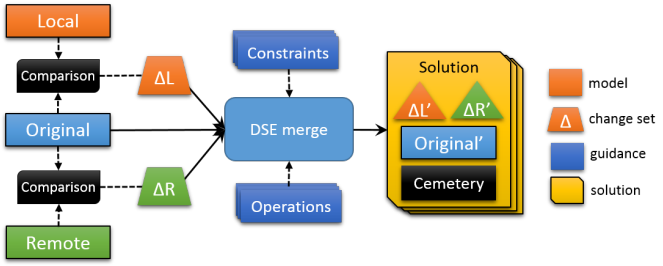

Fig. 3. Architecture of DSE Merge

(i) the well-formed merged model M; (ii) the set of non-executed changes $\Delta L'$,$\Delta R'$; and (iii) the collection of the deleted objects stored in $Cemetery$.

My contributions related to the fulfillment of *RQ-2* :

Contribution 2 I proposed a fine-grained property-based locking technique to avoid conflicts and an automated three-way model merge technique to resolve conflicts.

C2.1 *Fine-grained Property-based Locking.* I proposed a property-based locking technique as generalization of traditional fragment-based and object-based locking techniques which captures fine-grained locks as graph patterns and exploits incremental query engines to maintain and evaluate locks [35].

C2.2 *Automated Model Merge using DSE.* I proposed an automated three way model merge technique by adapting rule-based design space exploration to derive consistent and semantically correct merged models [36], [37].

C2.3 *Realization of DSE-merge.* I realized an infrastructure of automated model merge over EMF integrated into the Eclipse IDE [36], [32].

C2.4 *Evaluation.* I evaluated the scalability of the automated model merge and I compared the effectiveness of fine-grained property-based locking and traditional locking strategies for conflict prevention on a case study of offshore wind turbine controllers [36], [35], [33].

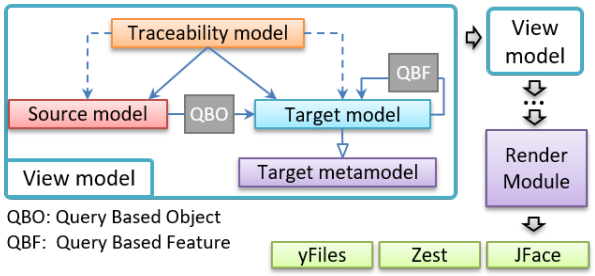

Fig. 4. Overview of integration architecture

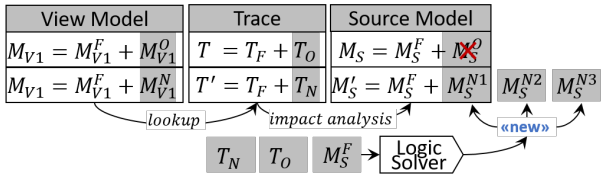

Fig. 5. Overview of backward change propagation

The novel concept of property-based locking has been carried out in a collaborative work [34] where my contribution is the first adaption in a practical setting.

Uniqueness. Our property-based approach is general and can be used for both implicit locking of subtrees and set of elements or explicit locking of a certain element and its incoming and outgoing references. In addition it extends these lock types with the definition of properties to provide less restrictive locking for the collaborators.

The closest to our merge approach are [19] and [20], but we rely on state-based comparison, apply a guided local-search strategy (vs. [20]), detect conflicts at runtime and allow complex generic merge operations (vs. [19]). Internally, we uniquely use incremental and change-driven transformations to derive the merged models. Finally, we reported scalability of merge process for models which are at least one order of magnitude larger compared to [19] and [20].

C. Synchronization of View Models

Approach. In [38], we introduced an approach where view models are conceptually equivalent to regular models and they are defined using a fully declarative, rule based formalism. *Preconditions* of rules are defined by graph patterns, which identify parts of interest in the source model. *Derivation rules* then use the match set of a graph pattern to define elements of the view model. Informally, when a new match of a query appears then the corresponding derivation rule is fired to create elements of the view model. When an existing match of a query disappears, the inverse of the derivation rule is fired to delete the corresponding view model elements.

View models derived by a unidirectional transformation are read-only representations, and they cannot be changed directly. To tackle this problem, we proposed an approach in [39] to automatically calculate possible source model candidates for a set of changes in different view models as depicted on Fig. 5. First, the possibly impacted partition of the source model is need to be identified by observing traceability links to restrict the impact of a view modification. Then the modified view models and the query-based view specification are transformed into logic formulae. Finally, multiple valid resolutions of the source model are generated using logic solvers corresponding to the changes of view models and the constraints of the source model from the users can manually select a proper solution.

My contributions related to the fulfillment of *RQ-3* :

Contribution 3 I proposed a novel technique of bidirectional synchronization of view models where the forward incremental synchronization is achieved by unidirectional derivation rules while the backward propagation of changes is generated using logic solvers.

C3.1 *Incremental Forward Synchronization.* I formalized a fully forward incremental, unidirectional synchronization technique of view models allowing chaining of views where the object of view model depend on the match set of the precondition of derivation rules [38], [40].

C3.2 *Change Impact Analysis.* I analyzed the impact of changes in underlying source models in case of backward propagation. The impacted part is added to the logic solver as additional constraints to calculate minimally modified source model candidates [39].

C3.3 *Realization of Forward Synchronization.* I realized the incremental and forward view synchronization technique where elementary derivation rules are captured by graph patterns and the reactive synchronization process uses the Viatra Event-driven Virtual Machine (EVM) [38].

C3.4 *Evaluation.* I evaluated the scalability of the proposed approaches on case studies from the avionics and the health-care domain [38], [39], [40].

The transformation of the preconditions described by graph patterns and the impacted parts to first order logic is the contribution of Oszkár Semeráth whereas my contributions are the impact analysis and the concept of using logic solver for backward propagation extended with impacted parts as additional constraints.

Uniqueness. Definition of a view model is *unidirectional*, while the forward propagation of the *operation-based* changes are *live*, *incremental* and executed *automatically* that also maintains *explicit traces*. At backward propagation, using partitioning as an additional input of the logic solver improves scalability issues and limits the impact of changes to a well-defined part of the source model.

ACKNOWLEDGEMENT

I would like to thank my advisor, Daniel Varro for his guidance during my research. I would also like to express my gratitude to Istvan Rath, Gabor Bergmann, Oszkar Semerath and Akos Horvath as well as Marsha Chechik, Fabiano Dalpiaz, Jennifer Horkoff and Rick Salay along with numerous colleagues and co-authors for sharing their ideas.

REFERENCES

[1] J. Whittle *et al.*, "The State of Practice in Model-Driven Engineering," *IEEE Software*, vol. 31, no. 3, pp. 79–85, 2014.

[2] Eclipse Foundation, "CDO," http://eclipse.org/cdo.

[3] ——, "EMFStore," http://eclipse.org/emfstore.

[4] Apache, "Subversion," https://subversion.apache.org/.

[5] G. Kramler *et al.*, "Towards a Semantic Infrastructure Supporting Model-based Tool Integration," in *GaMMa@ICSE'06*. ACM, 2006, pp. 43–46.

[6] Eclipse Foundation, "EMF Compare," http://eclipse.org/emf/compare/.

[7] ——, "EMF Diff/Merge," http://eclipse.org/diffmerge/.

[8] C. Clasen, F. Jouault, and J. Cabot, "VirtualEMF: A Model Virtualization Tool," in *Advances in Conceptual Modeling. Recent Developments and New Directions*, 2011, pp. 332–335.

[9] J. Tolvanen, "MetaEdit+ for Collaborative Language Engineering and Language Use (tool demo)," in *Tool Demo@SLE'16*, 2016, pp. 41–45.

[10] M. Maróti *et al.*, "Next Generation (Meta)Modeling: Web- and Cloud-based Collaborative Tool Infrastructure," in *MPM@MODELS'14*, 2014, pp. 41–60.

[11] Axellience, "Genmymodel."

[12] Obeo, "Obeo Designer," https://obeodesigner.com/en/collaborative-features.

[13] J. Gallardo *et al.*, "A Model-driven Development Method for Collaborative Modeling Tools," *J. Network and Computer Applications*, vol. 35, no. 3, pp. 1086–1105, 2012.

[14] E. Syriani *et al.*, "AToMPM: A Web-based Modeling Environment," in *Invited Talks, Demonstration Session, Poster Session, and ACM Student Research Competition@MoDELS'13*, 2013, pp. 21–25.

[15] K. Altmanninger *et al.*, "Amor–towards adaptable model versioning," in *MCCM@MoDELS'08*, vol. 8, 2008, pp. 4–50.

[16] F. Schwägerl *et al.*, "Model-based Tool Support for Consistent Three-way Merging of EMF Models," in *ACME@ECOOP'13*, 2013, pp. 2:1–2:10.

[17] J. Rubin and M. Chechik, "N-way Model Merging," in *ACM SIGSOFT Symp@ESEC/FSE'13*, 2013, pp. 301–311.

[18] P. Brosch *et al.*, "We can work it out: Collaborative Conflict Resolution in Model Versioning," in *ECSCW'09*, 2009, pp. 207–214.

[19] H. K. Dam *et al.*, "Inconsistency Resolution in Merging Versions of Architectural Models," in *WICSA'14*, 2014, pp. 153–162.

[20] U. Mansoor *et al.*, "MOMM: Multi-objective model merging," *Journal of Systems and Software*, vol. 103, pp. 423–439, 2015.

[21] H. Song *et al.*, "Instant and Incremental QVT Transformation for Runtime Models," in *MoDELS'11*, 2011, pp. 273–288.

[22] "Towards Automatic Model Synchronization from Model Transformations, author=Xiong, Yingfei and others, booktitle=ASE'07, pages=164–173, year=2007,."

[23] D. S. Kolovos, "Establishing Correspondences between Models with the Epsilon Comparison Language," in *ECMDA-FA'09*, 2009, pp. 146–157.

[24] A. Schurr, "Specification of Graph Translators with Triple Graph Grammars," in *Graph-Theoretic Concepts in Computer Science, WG'94*, 1994, pp. 151–163.

[25] OMG, "MOF 2.0 QVT."

[26] H. Gholizadeh *et al.*, "Analysis of Source-to-Target Model Transformations in QueST," in *Proceedings of the 4th Workshop on the Analysis of Model Transformations co-located with (MODELS 2015, Ottawa, Canada*, 2015, pp. 46–55. [Online]. Available: http://ceur-ws.org/Vol-1500/paper6.pdf

[27] D. Jackson, "Alloy Analyzer."

[28] H. Bruneliere *et al.*, "EMF Views: A View Mechanism for Integrating Heterogeneous Models," in *Conceptual Modeling - ER'15*, 2015, pp. 317–325.

[29] G. Bergmann, C. Debreceni, I. Ráth, and D. Varró, "Query-based Access Control for Secure Collaborative Modeling using Bidirectional Transformations," in *MoDELS'16*, 2016, pp. 351–361.

[30] C. Debreceni, G. Bergmann, I. Ráth, and D. Varró, "Deriving Effective Permissions for Modeling Artifacts from Fine-grained Access Control Rules," in *COMMitMDE@MoDELS'16*, 2016, pp. 17–26.

[31] ——, "Enforcing Fine-grained Access Control for Secure Collaborative Modeling using Bidirectional Transformations," *Software and System Modeling, MODELS 2016 Special Section*, 2017, submitted. [Online]. Available: https://goo.gl/ZAegbo

[32] C. Debreceni, G. Bergmann, M. Búr, I. Ráth, and D. Varró, "The MONDO Collaboration Framework: Secure Collaborative Modeling over existing Version Control Systems," *Tool Demo@ESEC/FSE'17*, 2017, in Press. [Online]. Available: https://goo.gl/uTsQeg

[33] A. Gómez, X. Mendialdua, G. Bergmann, J. Cabot, C. Debreceni, A. Garmendia, D. S. Kolovos, J. de Lara, and S. Trujillo, "On the Opportunities of Scalable Modeling Technologies: An Experience Report on Wind Turbines Control Applications Development," *ECMFA'17*, 2017, in Press. [Online]. Available: https://goo.gl/surozr

[34] M. Chechik, F. Dalpiaz, C. Debreceni, J. Horkoff, I. Ráth, R. Salay, and D. Varró, "Property-Based Methods for Collaborative Model Development," in *GEMOC+MPM@MoDELS'15*, 2015, pp. 1–7.

[35] C. Debreceni, G. Bergmann, I. Ráth, and D. Varró, "Property-based Locking in Collaborative Modeling," in *MoDELS'17*, 2017, in Press.

[36] C. Debreceni, I. Ráth, D. Varró, X. D. Carlos, X. Mendialdua, and S. Trujillo, "Automated Model Merge by Design Space Exploration," in *FASE'16*, 2016, pp. 104–121.

[37] H. Abdeen, D. Varró, H. A. Sahraoui, A. S. Nagy, C. Debreceni, Á. Hegedüs, and Á. Horváth, "Multi-objective Optimization in Rule-based Design Space Exploration," in *ASE '14*, 2014, pp. 289–300.

[38] C. Debreceni, Á. Horváth, Á. Hegedüs, Z. Ujhelyi, I. Ráth, and D. Varró, "Query-driven Incremental Synchronization of View Models," in *VAO@STAF'14*, 2014, pp. 31–38.

[39] O. Semeráth, C. Debreceni, Á. Horváth, and D. Varró, "Change Propagation of View Models by Logic Synthesis using SAT solvers," in *BX@ETAPS'16*, 2016, pp. 40–44.

[40] ——, "Incremental Backward Change Propagation of View Models by Logic Solvers," in *MoDELS'16*, 2016, pp. 306–316.