The MONDO Collaboration Framework: Secure Collaborative Modeling over Existing Version Control Systems

Csaba Debreceni*
Gábor Bergmann†

Márton Búr‡
István Ráth
Dániel Varró§¶
Budapest University of Technology and Economics
Hungary
debreceni,bergmann,bur,rath,varro@mit.bme.hu

ABSTRACT

Model-based systems engineering of critical cyber-physical systems necessitates effective collaboration between different stakeholders while still providing secure protection of intellectual properties of all involved parties. While engineering artifacts are frequently stored in version control repositories, secure access control is limited to file-level strategies in most existing frameworks where models are split into multiple fragments with all-or-nothing permissions, which becomes a scalability and usability bottleneck in case of complex industrial models.

In this paper, we introduce the MONDO Collaboration Framework, which provides rule-based fine-grained model-level secure access control, property-based locking and automated model merge integrated over existing version control systems such as Subversion (SVN) for storage and version control. Our framework simultaneously supports offline collaboration (asynchronous checkout-modify-commit) on top of off-the-shelf modeling tools and online scenarios (GoogleDocs-style short transactions) scenarios by offering a web-based modeling frontend.

Screencast Demo: https://youtu.be/Ix3CgmsYIU0

KEYWORDS

collaborative modeling, rule-based access control, locking, version control systems, automated model merge

ACM Reference format:
Csaba Debreceni, Gábor Bergmann, Márton Búr, István Ráth, and Dániel Varró. 2018. The MONDO Collaboration Framework: Secure Collaborative Modeling over Existing Version Control Systems. In *Proceedings of 11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Padeborn, Germany, September 04–08 (ESEC/FSE'17),* 5 pages.
https://doi.org/10.1145/3106237.3122829

1 INTRODUCTION

Model-based systems engineering has become an increasingly popular approach [15] in critical cyber-physical systems followed by many airframers or car manufacturers to simultaneously enhance quality and productivity. An emerging industrial practice of such system integrators is to outsource the development of various components to subcontractors in an architecture-driven supply chain. Collaboration between distributed teams of different stakeholders (system integrators, software engineers of component providers/-suppliers, hardware engineers, specialists, certification authorities, etc.) is intensified via the use of models.

In an *offline collaboration* scenario, collaborators check out an artifact from a version control system (VCS) and commit local changes to the repository in an asynchronous long transaction. In *online collaboration*, engineers may simultaneously edit a model in short synchronous transactions which are immediately propagated to all other users (similarly to online collaborative tools like Google Docs). Several collaborative modeling frameworks exist (CDO [12], EMFStore [13], GenMyModel [2], etc.), but security management is unfortunately still in a preliminary phase.

In fact, collaborative scenarios introduce significant challenges for security management, both in terms of confidentiality and integrity. For instance, the detailed internal design of a specific component needs to be revealed to certification authorities to obtain certification credit but they need to be hidden from competitors who might supply a different component in the system. Furthermore,

*Also with MTA-BME Lendület Research Group on Cyber-Physical Systems, Hungary.
†Also with MTA-BME Lendület Research Group on Cyber-Physical Systems, Hungary.
‡Also with MTA-BME Lendület Research Group on Cyber-Physical Systems, Hungary.
§Also with MTA-BME Lendület Research Group on Cyber-Physical Systems, Hungary.
¶Also with McGill University of Montreal, Canada.

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 and NSERC RGPIN-04573-16. The second author was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

ESEC/FSE'17, September 04–08, Padeborn, Germany

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/3106237.3122829

there may be critical aspects of the system model that may only be modified by domain experts having the appropriate qualifications.

Several industrial companies store and develop their system models in traditional VCSs (such as SVN) as *XMI*[1] files. However, these VCSs support access control on the granularity of files, projects or whole repositories. Thus fine-grained security goals may only be met by splitting the system model into many files, resulting in inflexible model fragmentation which becomes a scalability and usability bottleneck (e.g. over 1000 model fragments for automotive models).

Contributions. The MONDO Collaboration Framework[2] aims to (**C1**) extend traditional version control systems with advanced secure collaborative modeling features such as (**C2**) fine-grained model-level access control, (**C3**) property-based locking, (**C4**) automated model merge; in (**C5**) both offline and online collaboration scenarios.

C1. Integration with VCS: Our framework is currently integrated on the server side with Subversion (SVN) [1] (requested by the industrial partners of MONDO European FP7 Project [10]) as an underlying version control system, but the approach can be adapted to any other VCS supporting the appropriate hooks. In the offline scenario, users will interact with a version history in a regular SVN repository using an offline modeling tool and any SVN client. The results of online collaboration sessions will also be persisted in SVN.

C2. Fine-grained model-level access control: Traditional VCSs capture access control on the granularity of repositories or files, rigidly coupling model structure to permissions. Our framework can grant or deny access separately for each model element, cross-reference or attribute slot by offering fine-grained model access control policies (proposed in [3, 6, 11]) on top of traditional file-level access control. Since large industrial models may have millions of model elements, assigning explicit permissions individually to each element would be laborious; thus we provide a *rule-based access control policy language* [3, 6], where a single rule may grant or deny permissions for any number of assets. For concisely specifying and efficiently enforcing property-based access control rules, the affected assets are selected by a *declarative query*, which may take into account the type and attributes of model elements, as well as their wider context within the model. Our framework uses a bidirectional model transformation (a *lens* [3]) to enforce access rules. Models filtered according to read permissions may violate high-level, language-specific *well-formedness constraints*, but *internal consistency* is maintained so that they can always be persisted, loaded and traversed.

C3. Property-based locking: Effective collaboration requires that engineers working simultaneously do not interfere with each other. This is traditionally achieved by partitioning the model into fragment files and providing file-level locks; while object-level locks are also available in some systems [12]. The former is prone to overlocking, the latter to underlocking [5]. Our framework implements a novel property-based locking technique introduced in [4, 5] where locks protect operations by capturing their preconditions as a declarative query. The lock forbids other users from changing the model in a way that affects the result set of the query, thereby violating a precondition of the lock owner.

C4. Automated model merge: Asynchronous offline transactions lead to conflicts. Computing differences, conflicts and merged resolutions is significantly more complex over graph-based knowledge representations than over textual source code. Inter-model dependencies make conflicts surprisingly easy to introduce and hard to resolve. Our framework supports search-based automated model merge using a technique proposed in [7] that computes several candidate resolutions to a conflict thus reducing the required user interaction to simply choosing the preferred solution.

C5. Offline & online collaboration: Since the actual architecture of our framework has major differences between the offline and (web-based) online collaboration scenarios, the upcoming sections of the paper aim to highlight and explain these differences.

Added Value Compared to Related Tools. Compared to traditional file-based collaborative version control systems (e.g. SVN, Git), we provide an extra layer of fine-grained server-side access control and locking that flexibly decouple model hierarchy from permissions. Compared to model repositories (e.g. CDO[12], EMFStore[13], MetaEdit+[14]), our collaboration framework is transparent, i.e. it does not require any modifications to existing front-end (single-user) modeling tools.

Target Audience. Within the MONDO European FP7 Project [10], three kinds of users have been interacting with the MONDO Collaboration Framework. *Domain Engineers* read and write models taking advantage of advance support for access control, locking and merge. *Security Experts* act as superusers who are responsible for setting up secure repositories and defining access control policies (and lock types) that are enforced on Domain Engineers. Finally, *Language Developers* may extend our framework with knowledge specific to a given modeling language, which is required for the server to enforce fine-grained write access control, and optional for the client to improve the merge process.

2 OFFLINE COLLABORATION TOOL

In the offline scenario, models are stored in a Version Control System (VCS) as files (e.g. *XMI*, *binary*) and users work on local working copies of the models in long transactions called commits. The goal of our approach is to enforce fine-grained model access control rules and locks on top of existing security layers available in the VCS such as SVN or Git.

In the offline case, the MONDO Collaboration Framework uses two types of repositories called *gold* and *front* depicted in Fig. 1. The *gold* repository contains complete information about the models, but it is not accessible to collaborators. Each user has a *front* repository, containing a full version history of filtered and obfuscated copies of the gold models. A user first submits a new model version to his/her front repository; then changes introduced in this revision will be interleaved into the gold model; finally, the new gold revision will be propagated to the front repositories of other users. As a result, each collaborator continues to work with a

[1] XMI format http://www.omg.org/spec/XMI/
[2] Source code of the framework can be found at the following link: https://github.com/FTSRG/mondo-collab-framework

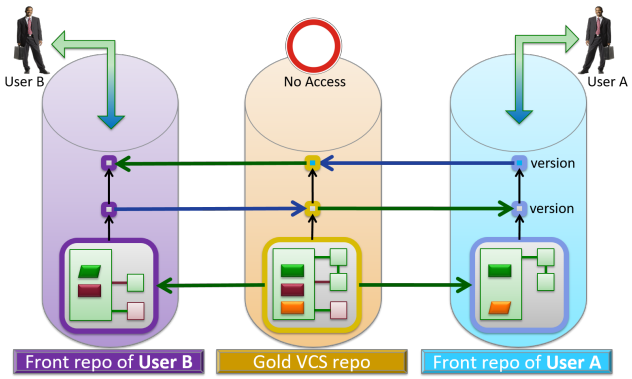

Figure 1: MONDO Offline Collaboration - Architecture

dedicated VCS repository as before, being unaware that this front repository may contain filtered and obfuscated information only.

Key Features

Secure Access Control. Security policies[6] consist of fine-grained write and read access control rules. *Write access control* rules limit which model elements of a model are allowed to be changed, while *read access control* rules determine which model elements of a model are visible to a user. These rules can be defined on the language and the model level.

Access control rules are enforced upon each commit at a front repository[3]. When a new revision is attempted to be added to a front repository, its success is conditional to validating the write access control rules in the gold repository. If the commit is accepted, the read access control rules will determine which changes will be propagated to the other front repositories. This scheme enforces the access control rules even if users access their personal front repositories using standard VCS clients and off-the-shelf modeling tools, greatly reducing adoption costs in the offline scenario.

In practice, access control is enforced by *hooks* of the repositories. Upon each commit, the *pre-commit hook* of the front repository applies the *security lens* (a bidirectional model transformation parameterized by the user identity, the access control policy and the locks) to obtain the updated gold model, and rejects the commit immediately if any write rule is violated. If the commit is accepted, the gold repository is updated accordingly, where a *post-commit hook* will apply the lens to propagate changes to the rest of the front repositories.

The lens needs to identify corresponding objects between the models in front and gold repositories. By default, simple global identifiers are used to establish object identities, but *Language Developers* are able to extend this behavior to find an identifier-free solution.

Access control policies for different collaborators are defined by security experts who are superusers from a security perspective. We made the design decision that the model-level *authorization files* are stored and versioned in the same VCS as the models. Thus policy files may evolve naturally along with the evolution of the contents of the repository. Since the rule names and parameters do not normally contain sensitive information, these policies may potentially be made readable for every user, without major security concern. However, if it is required, the readablility of these *authorization files* can be denied separately on each front repository. As future work to further improve usability with tighter feedback, offline clients may approximatively evaluate write access rules on their (incomplete) offline copies themselves prior to committing.

Automated Model Merge. When concurrent modifications are committed to the VCS, different versions of the models need to be merged. During the merge phase, changes introduced by different collaborators may contradict each other which leads to conflicts that have to be resolved. The MONDO Collaboration Framework integrates the automated model merge approach proposed in [7] into the Eclipse teamwork UI. The tool computes and displays all possible candidate resolutions of a conflict, and the user can select the most suitable one. Choosing a solution is assisted by several metrics such as the number of elements deleted, number of changes included in the solution, etc.

By default, the model merge feature contains elementary conflict resolution strategies e.g. modify vs. delete objects that can be extended with complex domain-specific resolution rules by *Language Developers*.

Locking. Locks reduce the need for merging by preventing other users from modifying the models in a way that would cause conflicts. Similarly to security policies, fine-grained locks[5] can be defined as rules on the levels of metamodel and instance model in accordance with [4]. The evaluation of locks also happens at commit time at the gold repository, and its actual implementation is similar to checking access control rules. As a potential future improvement, certain locks could be checked offline, just like write access rules.

Off-the-shelf Client. The *front* repository acts as a standard VCS server, so users can use their favorite off-the-shelf VCS clients and off-the-shelf (single-user) modeling tools without any mandatory collaboration-specific customization. This opt-in compatibility greatly enhances usability in case of the offline scenario.

Management UI. An optional client interface is integrated into the Eclipse IDE to allow users to discover the address of their front repository, while superusers can also execute server-side tasks, e.g. reset front repositories.

3 ONLINE COLLABORATION TOOL

In the online scenario, several users can join a *collaborative modeling session* (a *whiteboard*, see Fig. 2), and simultaneously display and edit the same model with short transactions where changes are propagated immediately to other users. Models are kept in server memory and users access the model directly on the server, in contrast to the offline scenario, where users manipulate local copies of the models.

Similarly to modern collaborative editing tools (such as Google Sheets [9]), whiteboards can be operated transparently: whenever the first user attempts to open a given model, a new whiteboard is started; subsequent users opening the model will join an existing whiteboard. When all users have left, the whiteboard can be

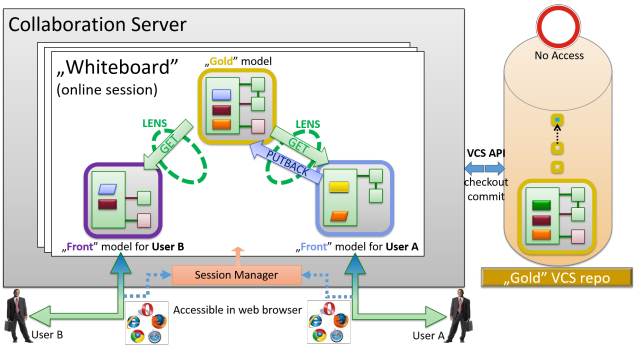

Figure 2: Overview of Online Collaboration

disposed. The model may be persisted periodically, or on demand ("save button") to a VCS. The *session manager* component enables collaborators to start / join or leave whiteboard sessions, persist models to disk and VCS. When a new whiteboard session is initiated, file-level locks are placed on the resources of the related model to prevent conflicts in the VCS upon persisting.

Collaborators perform model edit operations via a client application that directly connects to the server and the whiteboard in particular. In the current solution, this client is a modeling tool implemented as a web application accessible by any modern web browser (see Fig. 3) where the modeling tool can be extended by any known Eclipse plug-in. In the future, we plan to provide an accessible web API (and/or reuse CDO [12] API) to support desktop IDEs with our online features.

Key features

Secure Access Control and Locking Our framework enforces read access control rules for different collaborators also for the online scenario, thus different users do not actually see the same model contents. Instead, each of them actually connects to a dedicated view of the model that has been filtered and obfuscated, but kept in synch with (visible) modifications performed by other users.

The security lens, run as a live transformation, is responsible for incrementally maintaining consistency between front and gold models after each modification by enforcing access control rules and preventing lock violations. It is worth highlighting that access control and locking policies are editable by users with sufficient privileges and they will be reapplied upon modification within the same session.

Automated Model Merge In online collaboration, automated model merge is irrelevant: only short transactions (single edit operations) are allowed which are propagated immediately to all other views (i.e. model edit is synchronous), thus model conflicts may occur only in a very short time window.

Undo/Redo Undo/Redo is supported for all the changes carried out by a user. Each client stores only their own change history, thus no one can revert changes carried out by other users.

User Interface The prototype user interface of our tool, depicted in Fig. 3, uses the editors automatically generated from EMF metamodels[3] which also provides similar modeling environment as the desktop Eclipse IDE. By default, it provides (1) *Tree editors* for accessing the models and (2) *Properties* view to modify the attributes of model element. We also implemented several additional views such as (3) *Current User* which shows the currently logged in user, and (4) *Model Explorer* that lists the accessible models, policy definitions (*.macl), lock definitions (*.mpbl) and related query files in the underlying VCS.

4 INITIAL EXPERIMENTAL EVALUATION

A usability evaluation of MONDO Collaboration Framework has been carried out [8] by industrial partners *IK4-Ikerlan* and *UNINOVA* within the evaluation phase of MONDO project in the context of (1) a wind turbine case study (with small models but many users) and (2) a building information model (with models over 100,000 elements but fewer users). The number of concurrent users working on different views of the same model and the time for propagating changes and notifications among the concurrent users were evaluated both in offline and online cases, and the engineers could successfully collaborate in both cases.

Moreover, we also carried out an initial scalability evaluation of the online collaboration scenario over a synthetic domain with model generator in [3]. We gradually increased the number of concurrent users to 100 and the size of the view models (to ca. 2300 objects, 3100 references) and the average propagation of 100 changes was still less then 0.25 sec. Moreover, execution time was dependent on the size of the change but not the sheer size of the model.

[3]EMF and RAP integration: http://tinyurl.com/emf-rap-integration

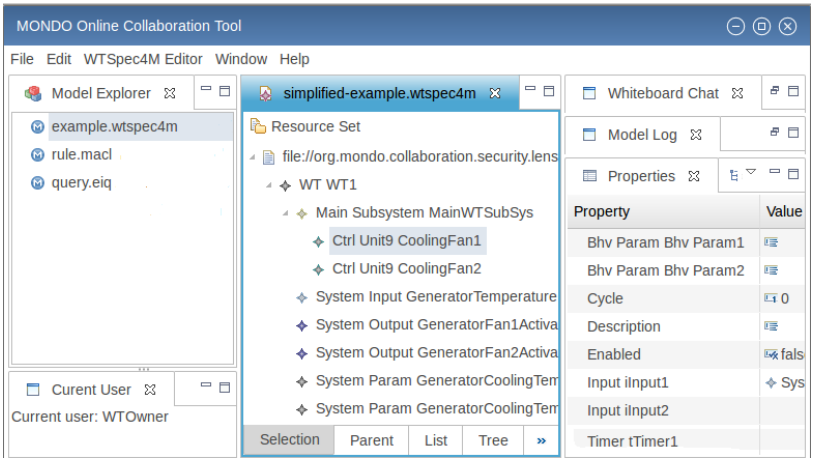

Figure 3: User Interface for Online Collaboration Tool

REFERENCES

[1] Apache. 2017. Subversion. https://subversion.apache.org/.

[2] Axellience. 2017. GenMyModel. http://www.genmymodel.com.

[3] Gábor Bergmann, Csaba Debreceni, István Ráth, and Dániel Varró. 2016. Query-based access control for secure collaborative modeling using bidirectional transformations. In *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, Saint-Malo, France, October 2-7, 2016*. 351–361. http://dl.acm.org/citation.cfm?id=2976793

[4] Marsha Chechik, Fabiano Dalpiaz, Csaba Debreceni, Jennifer Horkoff, István Ráth, Rick Salay, and Dániel Varró. 2015. Property-Based Methods for Collaborative Model Development. In *Joint Proceedings of the 3rd International Workshop on the Globalization Of Modeling Languages and the 9th International Workshop on Multi-Paradigm Modeling co-located with ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems, GEMOC+MPM@MoDELS 2015, Ottawa, Canada, September 28, 2015*. 1–7. http://ceur-ws.org/Vol-1511/paper-01.pdf

[5] Csaba Debreceni, Gábor Bergmann, István Ráth, and Dániel Varró. Property-based Locking in Collaborative Modeling. In *ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS 2017)*. In Press.

[6] Csaba Debreceni, Gábor Bergmann, István Ráth, and Dániel Varró. 2016. Deriving Effective Permissions for Modeling Artifacts from Fine-grained Access Control Rules. In *Proceedings of the 1st International Workshop on Collaborative Modelling in MDE (COMMitMDE 2016) co-located with ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2016), St. Malo, France, October 4, 2016*. 17–26. http://ceur-ws.org/Vol-1717/paper6.pdf

[7] Csaba Debreceni, István Ráth, Dániel Varró, Xabier De Carlos, Xabier Mendialdua, and Salvador Trujillo. 2016. Automated Model Merge by Design Space Exploration. In *Fundamental Approaches to Software Engineering - 19th International Conference, FASE 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*. 104–121. https://doi.org/10.1007/978-3-662-49665-7_7

[8] Abel Gómez, Xabier Mendialdua, Gábor Bergmann, Jordi Cabot, Csaba Debreceni, Antonio Garmendia, Dimitris S. Kolovos, Juan de Lara, and Salvador Trujillo. 2017. On the Opportunities of Scalable Modeling Technologies: An Experience Report on Wind Turbines Control Applications Development. In *13th European Conference on Modelling Foundations and Applications*. Springer, Marburg, Germany.

[9] Google. 2017. Google Docs. http://www.docs.google.com/.

[10] Dimitrios S. Kolovos, Antonio García-Domínguez, Richard F. Paige, Esther Guerra, Jesús Sánchez Cuadrado, Juan de Lara, István Ráth, Dániel Varró, Gerson Sunyé, and Massimo Tisi. 2016. MONDO: Scalable Modelling and Model Management on the Cloud. In *Joint Proceedings of the Doctoral Symposium and Projects Showcase at STAF 2016*. 55–64.

[11] Salvador Martínez Perez, Jokin García, and Jordi Cabot. 2016. Runtime support for rule-based access-control evaluation through model-transformation. In *Proceedings of the 2016 ACM SIGPLAN International Conference on Software Language Engineering, Amsterdam, The Netherlands, October 31 - November 1, 2016*. 57–69. http://dl.acm.org/citation.cfm?id=2997375

[12] The Eclipse Foundation. 2017. CDO. http://www.eclipse.org/cdo.

[13] The Eclipse Foundation. 2017. EMFStore. http://www.eclipse.org/emfstore.

[14] Juha-Pekka Tolvanen. 2007. MetaEdit+: Domain-Specific Modeling and Product Generation Environment. In *Software Product Lines, 11th Int. Conf. SPLC 2007, Kyoto, Japan*. 145–146.

[15] Jon Whittle, John Hutchinson, and Mark Rouncefield. 2014. The State of Practice in Model-Driven Engineering. *IEEE Software* 31, 3 (2014), 79–85. https://doi.org/10.1109/MS.2013.65