EDITOR INFO.

SECTION TITLE. Collaborative Modeling in Software Engineering

ARTICLE TITLE. Secure Views for Collaborative Modeling

AUTHOR.
Csaba Debreceni[1,2], Gábor Bergmann[1,2,5], István Ráth[1,3], Dániel Varró[1,2,4]
AUTHOR AFFILIATION.
[1]Budapest University of Technology and Economics, Hungary
[2]MTA-BME Lendület Research Group on Cyber-Physical Systems, Hungary
[3]IncQuery Labs Ltd., Hungary
[4]McGill University, Canada
[5] supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences

ABSTRACT. Model-based systems engineering necessitates effective collaboration between different collaborators, teams, and stakeholders. Traditional approaches used for managing concurrent code-based development do not naturally extend to collaborative modeling, which implies novel challenges. We present a collaborative modeling framework that provides secure views with precisely defined model access to each collaborator by rule-based model-level access control policies.

CHALLENGES

The adoption of model-based systems engineering (MBSE) by system integrators (like airframers or car manufacturers) has been steadily increasing in the recent years [1]. MBSE enables to detect design flaws early and generate various artifacts (source code, configuration tables, etc.) automatically from high-quality system models. As a common industrial practice, system integrators frequently outsource the development of various components to subcontractors in an architecture-driven supply chain where the collaboration between cross-organizational teams is facilitated by sharing models stored in model repositories [2]. However, effective collaboration is hindered by numerous factors.

Protection of intellectual property over heterogeneous teams. Cross-organizational collaboration introduces significant challenges for access control management to protect the respective Intellectual Property (IP) of different parties. For instance, the detailed internal design of a component needs to be revealed to certification authorities, but it needs to be hidden from competitors who might supply a different component in the system. Furthermore, certain critical aspects of the system model may only be modified by domain experts with appropriate qualifications. Due to the lack of support for cross-company collaboration in existing modeling repositories, very strict infrastructure-level security policies are in place at companies, which prevent effective collaboration.

Model fragmentation and lack of fine-grained access control. In current industrial MBSE practice, system models are persisted in repositories such as (1) Version Control Systems (VCS), e.g. SVN or Git, (2) local database backends (e.g. in MetaEdit+ [3]), (3) dedicated object-based storages such as CDO (e.g. in Team for Capella), or (4) cloud-based solutions (e.g. NoMagic Teamwork Cloud).

Unfortunately, access control management is still in a preliminary phase in these repositories as they support permission assignments only globally, or using *fragments* (i.e.

on the level of files or projects). In order to share *parts of* system models between collaborators, models need to be split into a large number of static fragments (e.g. over 1000 for automotive models). Consequently, the re-fragmentation of the model, which may be necessary when roles, policies or models evolve, is hard or infeasible. Therefore, static fragmentation becomes both a scalability and usability bottleneck.

Static fragmentation can be mitigated by *fine-grained access control* where each model element may have its own set of permissions. Unfortunately, large industrial models may have millions of model elements, thus explicitly assigning permissions for each element would be labor-intensive and error-prone. Moreover, understanding and maintaining permissions after model changes can be problematic.

Online and offline collaboration. System models are traditionally developed either in an offline *or* online manner. In *offline collaboration*, engineers check out an artifact from a repository into a local copy and commit local changes to the repository in asynchronous (long) transactions. In *online collaboration*, engineers may simultaneously edit a model in short synchronous transactions which are immediately propagated to all other users. This strategy is similar to online collaborative office tools like Google Docs, and it is showcased in modeling tools like WebGME [4], ATOMPM [5], GenMyModel [6] or MetaEdit+ [3].

In case of access control restrictions, certain model changes should not be propagated to some collaborators while certain model modifications may be disallowed and rejected. In the offline case, all information available to a specific user needs to be provided as a self-contained model that can be displayed and edited by existing off-the-shelf modeling tools. Hiding elements in a modeling tool is insufficient, as the IP is still accessible on the client side e.g. by file inspection. Hence, a filtered but modifiable model that excludes any confidential information needs to be sent to each client. Furthermore, the online scenario requires immediate change propagation where model modifications need to be evaluated in an efficient, reactive way in response to the change.

Integration with existing techniques and tools. Effective access control for collaborative modeling should smoothly integrate with other collaboration practices such as *version histories*, *user authentication*, *lock management* (to temporarily prevent certain model edits) or *conflict resolution* by differencing and merging. While traditional VCS frameworks provide efficient support for handling text-based design artifacts, their model-level counterparts require sophisticated techniques. Furthermore, the seamless integration of a collaboration layer with existing toolchains is a key industrial need.

SECURE VIEWS FOR RULE-BASED MODEL-LEVEL ACCESS CONTROL

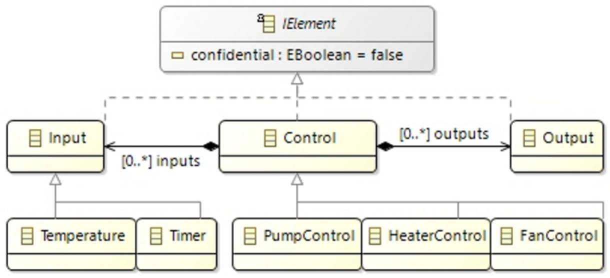
To address these challenges, we have developed the MONDO COLLABORATION FRAMEWORK that enhances collaborative modeling repositories by providing *secure views* as an extra protection layer with rule-based model-level access control simultaneously enforced for both offline and online collaboration scenarios and adapted to software configuration management.

Model-level Access Control Rules

Security policies define *access control rules* that grant or deny read or write permissions over some *assets* (objects, attributes, references) for certain collaborators. Individual assets are read-only if write permissions are denied and they are hidden if read permissions are denied. Complex *attribute-based access control* strategies can be implemented by combining access control rules with graph queries. *Graph queries* provide an expressive *declarative language* (from the VIATRA [7, 8] open source project) to select which assets to restrict. *Access control rules* may reuse graph queries to define complex hierarchies and dependencies between individual assets based on the current state of the model.

Access control rules are illustrated on a small example, extracted from the domain of offshore wind turbines [9] which was a case study of the MONDO Project [10]. A *control unit* is responsible for a certain type of physical device (e.g. pump, heater, fan). It may depend on multiple *input*s (e.g. temperature) and produce multiple *output*s (see metamodel in Figure 1, upper-left part). Each control unit requires special domain knowledge, thus a specialist called *IOManager* needs to oversee the inputs and outputs.

The security policy (Figure 1, right part) provides access to non-confidential inputs and outputs only. By default, it denies access to any asset in the model. Here all access control rules use the same graph query *objectIO* (Figure 1, lower-left part) that selects all *inputs* and *outputs* and the value of their *confidential* attribute. Rule *enableIO* grants read and write permissions to all objects selected by graph query *objectIO*. Rule *disableConfidentialAttr* overrides this to make the *confidential* attribute of the same objects tamper-proof. Finally, rule *disableConfidentialIO* takes those inputs and outputs where the query returns *true* for *isConfidential*, and entirely hides them from subcontractors (members of the *Externals* group including IOManagers). Note that high-priority rules override lower-priority rules as well as the default permissions.



```
pattern objectIO(obj : IElement, isConfidential) {
  Input.confidential(obj, isConfidential);
} or {
  Output.confidential(obj, isConfidential);
}
```

```
policy deny RW by default {

  rule enableIO allow RW to IOManager {
    from query objectIO
    select Object(obj)
  } with priority 1

  rule disableConfidentialAttr deny W to IOManager {
    from query objectIO
    select Attribute(obj, confidential, isConfidential)
  } with priority 2

  rule disableConfidentialIO deny RW to Externals {
    from query objectIO
    where isConfidential bound to true
    select Object(obj)
  } with priority 3

}
```

FIGURE CAPTION. Figure 1. Sample access control policy

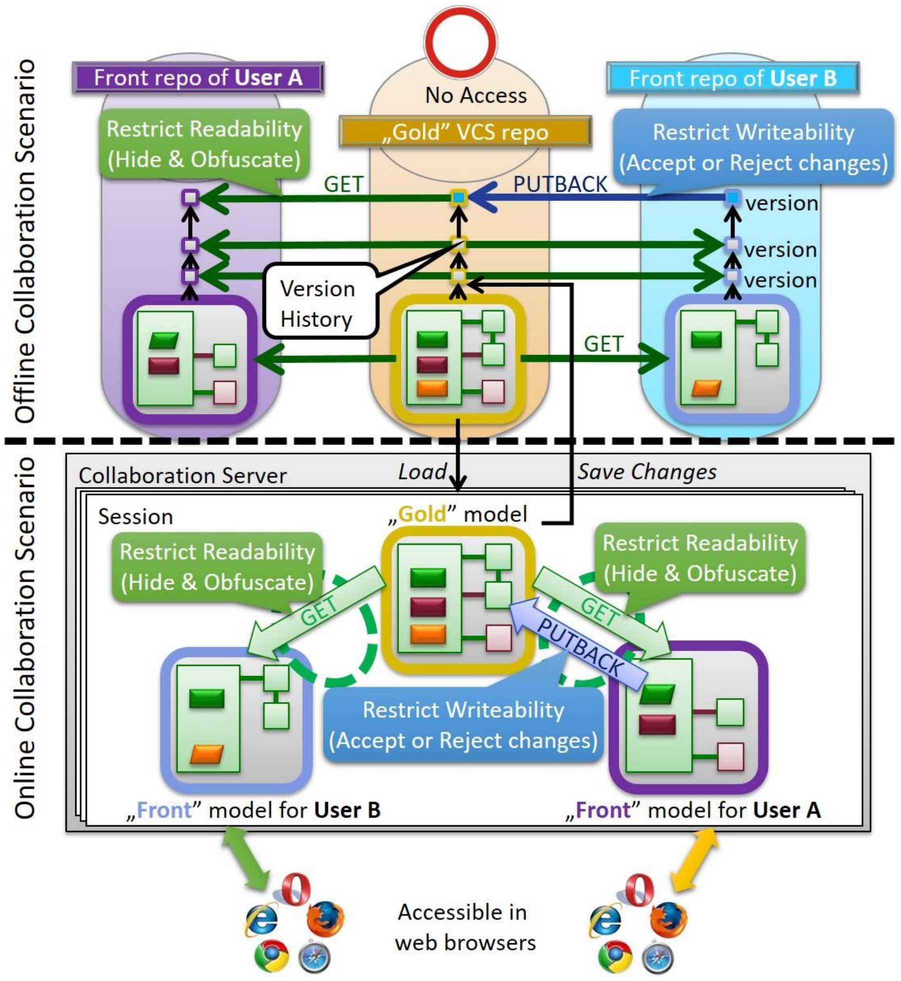
Enforcing Access Control Rules in Secure Online and Offline Collaboration

Our approach assumes the existence of a *gold* model that includes all information and IP, but it is not directly accessible to collaborators. Instead, access control rules are enforced by providing *secure views* (called *front* models) which contain only those model elements that are accessible to a specific collaborator. When collaborators modify their front models, the changes are propagated back to the *gold* model and then to other collaborators.

Such synchronization between updatable secure views can conceptually be described by a *lens* [11], which is a pair of model transformations GET and PUTBACK. GET is responsible for read access control by filtering the gold model (hiding assets) into the front model according to read permissions. PUTBACK is responsible for checking modifications of the front model against write permissions and propagating them to the gold model if accepted.

For *secure offline collaboration* [12] (Figure 2, upper part), each collaborator gets a dedicated *front repository* that contain the full version history of a *front* model. Similarly, a *gold repository* contains the version history of the *gold* model with complete information. Each collaborator continues to use an existing off-the-shelf client to access the front repository (but not the gold repository). Hence there is infrastructure-level protection against unauthorized model access. When a collaborator uploads some modifications to her front repository, these changes are propagated to the gold model by a PUTBACK operation. Then a new entry is added to version history of the gold repository, and all other front repositories are synchronized by GET operations where those changes are visible.

For *secure online collaboration*, several users can join a shared modeling session (Figure 2, lower part) to simultaneously display and edit the same model where changes are propagated immediately to other users. The gold model is loaded to server memory from the gold repository of the underlying (offline) repository. When a collaborator joins the session, a dedicated front model is derived from the gold model. Collaborators access their front model directly on the server (e.g. using a web browser), in contrast to the offline scenario, where users manipulate local copies of the models. As such, sensitive and restricted information is always kept on the servers, and it never reaches the modeling clients. Our lens transformations [13] operate in an incremental way: if a front model is modified, then the resulting PUTBACK and the subsequent GETs (propagating to other collaborators) act only on the changes. Therefore, after the initialization phase when the gold and front models are loaded into the memory, execution time is only proportional to the size of the changes and not to the size of the model.



Offline Collaboration Scenario
Front repo of User A
No Access
Front repo of User B
Restrict Readability (Hide & Obfuscate)
„Gold" VCS repo
Restrict Writeability (Accept or Reject changes)
GET
PUTBACK
version
version
version
Version History
GET
Online Collaboration Scenario
Collaboration Server
Load
Save Changes
Session
„Gold" model
Restrict Readability (Hide & Obfuscate)
Restrict Readability (Hide & Obfuscate)
GET
GET
PUTBACK
Restrict Writeability (Accept or Reject changes)
„Front" model for User B
„Front" model for User A
Accessible in web browsers

FIGURE CAPTION. Figure 2. Architecture of offline and online collaboration

Secure Software Configuration Management

Secure views can be directly used in industrial *software configuration management* (SCM) scenarios.

Change requests. In SCM, changes may require the approval of multiple collaborators in *model/code reviews*. Without sufficient permissions to commit to gold, a *proposer* may create a *change request*. The request can be inspected for approval by other collaborators, and committed once it is *signed off* by one or more reviewers with *jointly sufficient* write privileges.

Read access is enforced on change requests via GET before being shown to reviewers. However, as access control is attribute-based, this filtering can be manipulated, e.g. an attacker may propose changes in artifact ownership. Privilege escalation is averted as follows: first, a modified PUTBACK applies those proposed changes that are allowed with the joint write permissions of the proposer and the reviewer; then the merged result model is filtered by GET for the reviewer. This way, the secure view on the change request will only contain modifications that the reviewer can enact, or the proposer would be allowed to commit anyway.

Collaboration workflows. In the industrial context, complex collaboration workflows along multiple branches (production, feature, etc.) are needed to co-evolve multiple versions of a system design for continuous engineering. Since access control rules use a graph query language for identifying assets, the scope of restrictions can be easily extended to support multiple branches. Domain-specific types defined by the metamodel are complemented with SCM-specific concepts like workspace, resource, branch, revision or tag, but no further adaptation is needed to the access control language. Thus one can specify access control policies where a collaborator is restricted to observe model versions on specific branches or created after certain revisions.

Industrial applications. Together with the industrial partners of MONDO, we carried out a detailed scalability evaluation of the collaboration server with respect to increasing number of front repositories, increasing size of models or increasing change size (for the offline case) with promising results [9, 13, 12]. Further practical relevance stems from the fact that certain industrial products (like NoMagic Teamwork Cloud) have already adapted a mixed online and offline collaboration architecture, thus the concepts of secure views are directly applicable.

While we promote graph queries to drive rule-based access control management for collaborative modeling, it is worth emphasizing that graph queries can also support other server-side tasks such as change impact analysis or automated conformance checks. Moreover, since lenses for checking access control rules can be wrapped into micro-services (like Docker), we foresee the seamless adaptation and integration of secure views into existing product line management tools or future collaborative cloud-based model repositories.

Limitations. Our collaboration server is currently integrated with Subversion (SVN) as requested by the industrial partners of MONDO European FP7 Project, where version history is managed in a standard SVN repository and users collaborate by using existing modeling tools and any SVN client. Integration with Git as underlying VCS technology is ongoing work.

PRACTICAL BENEFITS

Key benefits of our collaborative modeling framework for MBSE include the following:

Collaboration of heterogeneous stakeholders: Our framework supports collaborative modeling between engineering teams of different companies (e.g. an integrator and its subcontractors) while protecting their intellectual property with the secure storage of the gold model.

Extra layer of access control: Model-level fine-grained access control using secure views injects an extra layer of protection on top of existing protection offered by the underlying repository.

Validation of access control policies: Access control rules support the consistent assignment and maintenance of permissions for large models and enable the systematic validation of access control policies (e.g. to ensure export control regulations or investigate a security breach).

Compliance with SCM practices: Access control policies can be defined for modern SCM practices to collaborate along multiple branches, formal change request, etc.

Smooth integration with existing tools: Our framework extends existing server-side repositories while keeping client-side modeling tools intact, thus engineers may continue using existing collaborative tools.

As such, powerful existing collaboration practices used in software engineering can be complemented when collaborating over models.

References

1. Whittle, J. et al., The state of practice in model-driven engineering, IEEE Software 31(3) (2014):79-85.
2. Di Rocco, J. et al., Collaborative repositories in model-driven engineering, IEEE Software 32(3) (2015):28-34.
3. Tolvanen, J-P. et al, Model-Driven Development Challenges and Solutions - Experiences with Domain-Specific Modelling in Industry, MODELSWARD'16 (2016):711-720.
4. Maróti, M. et al., Next Generation (Meta) Modeling: Web-and Cloud-based Collaborative Tool Infrastructure, MPM@ MoDELS'14 (2014): 41-60.
5. Syriani, E. et al., ATOMPM: A web-based modeling environment., MODELS'13 (2013):21-25.
6. Dirix, M. et al., GenMyModel: an online UML case tool, ECOOP, 2013.
7. Eclipse Project, VIATRA https://www.eclipse.org/viatra/.
8. Varró, D. et al., Road to a reactive and incremental model transformation platform: three generations of the VIATRA framework, Software and System Modeling 15(3) (2016):609-629.
9. Gómez, A. et al., On the Opportunities of Scalable Modeling Technologies: An Experience Report on Wind Turbines Control Applications Development, ECMFA'17 (2017):300-315.
10. Bagnato, A. et al., Flexible and scalable modelling in the MONDO project: Industrial case studies., XM@ MoDELS'14 (2014):42-51.
11. Foster, J. et al., Updatable Security Views, 22nd IEEE Computer Security Foundations Symposium (2009):64–70.
12. Debreceni, C. et al., Enforcing Fine-grained Access Control for Secure Collaborative Modeling using Bidirectional Transformations, Software and Systems Modeling (2017),DOI:10.1007/s10270-017-0631-8.
13. Bergmann, G. et al., Query-based Access Control for Secure Collaborative Modeling using Bidirectional Transformations, MoDELS'16 (2016):351-361.