ReMoDD Eclipse Plug-in: Collaborative Modeling Using a Model Repository

Mohammed Al-Refai[1], Andrew Jacobson[1], Sudipto Ghosh[1], James M. Bieman[1], Betty H. C. Cheng[2]
[1] Computer Science, Colorado State University, Fort Collins, CO, USA
Email: al-refai,akj,ghosh,bieman@cs.colostate.edu
[2]Computer Science and Engineering, Michigan State University, East Lansing, MI, USA
Email: chengb@cse.msu.edu

***Abstract*—The Repository for Model-Driven Development (REMODD) is a community resource developed to support the research and education activities of researchers and educators in the Model-Driven Development (MDD) community. Researchers and practitioners can use the repository as a vehicle for sharing exemplar models, illustrative descriptions of modeling methodologies and techniques, detailed modeling case studies, modeling success stories, and other forms of modeling experience and knowledge. Recent extensions to REMODD support the seamless integration of REMODD into existing modeling frameworks, thereby improving the access to REMODD artifacts. In particular, we have developed an Application Programming Interface (API) to enable other repositories and existing modeling tools to directly access REMODD search, browsing, and retrieval facilities. We validate and demonstrate the API utility through an Eclipse plug-in that allows a user to collaboratively work on modeling artifacts that are available in REMODD.**

***Index Terms*—collaborative modeling, Eclipse plug-in, MDE, ReMoDD, version control**

***Link to Video Tutorial*—**http://remodd.org/v1/plugin-demo

I. INTRODUCTION

The Repository for Model Driven Development (REMODD)[1] contains artifacts that researchers in the *Model-Driven Development* (MDD) community can use to guide and validate their research. REMODD artifacts include detailed MDD case studies describing the use of models (including graphical models, such as UML and Simulink, as well as models expressed in formal specification languages such as Z, Alloy, and Statecharts) throughout the development lifecycle, source code describing implementations of design models, examples of models reflecting good and poor modeling practices, benchmark models that can serve as the basis for evaluating model manipulation techniques, reusable model transformations, and pedagogical materials that can enhance the teaching of MDD concepts. The intent for REMODD is to provide a community resource for storing MDD artifacts that can be used to gain significant insights into the use of models across the software lifecycle, as a source of data for MDD experiments, as a source of models for testing MDD tools, and to better understand relationships among ongoing MDD research projects. In particular, educators in academia and industry can use REMODD resources to illustrate modeling concepts and approaches in the classroom.

We expect that the artifacts in REMODD will evolve as researchers and educators use them in their work. Researchers may need new versions of existing models to support experimentation in studying approaches involving model evolution and testing. Educators may require their students to extend models to teach concepts in refactoring. The new versions of models will need to be stored in REMODD to support future research and education. Previously REMODD[1] supported versioning capabilities via Drupal but did not support access control and configuration management for simultaneous collaborative modeling activities by contributors modifying the same artifact. Modifications to artifacts are typically performed in development environments that support modeling tools, which require the availability of tools and Application Programming Interfaces (API) to access REMODD, check out artifacts, and push and commit changes to the artifacts.

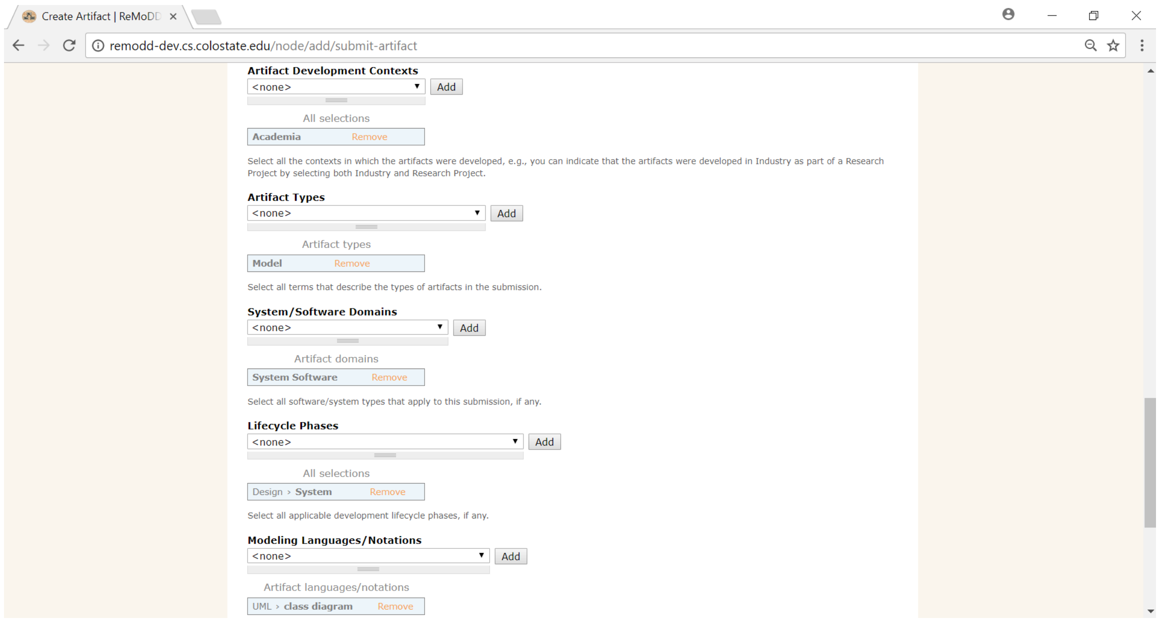
In this paper we describe key elements of the REMODD API and illustrate its use with the Papyrus Modeling environment[2] and our REMODD plug-in. In the video tutorial, we demonstrate several usage scenarios that are described below.

II. USAGE SCENARIOS

We developed an API for REMODD that defines the Java interfaces, classes, and exceptions necessary to support (1) user authentication, and login and logout processes, (2) new artifact creation and submission, (3) search and browse, (4) retrieving artifacts from REMODD, (5) modifying retrieved artifact meta-data and content, and (6) pushing a modified artifact back to REMODD. In order to facilitate access and support configuration management, REMODD now makes use of the `git` platform. We demonstrate the utility of the API using an Eclipse plug-in. Usage scenarios of the plug-in are described below and demonstrated in the video tutorial.

Scenario-1: This scenario takes the user through the artifact creation process (i.e., AirlineSystem artifact), using REMODD's web interface starting from the REMODD homepage. The user can be the artifact's author or someone uploading the artifact on behalf of the author. After providing the author's information and brief description of the artifact as shown in Figure 1, the user submits the candidate artifact for inclusion in REMODD.

[1]http://www.remodd.org/
[2]https://eclipse.org/papyrus/



Artifact Development Contexts
<none> Add
All selections
Academia Remove
Select all the contexts in which the artifacts were developed, e.g., you can indicate that the artifacts were developed in Industry as part of a Research Project by selecting both Industry and Research Project.

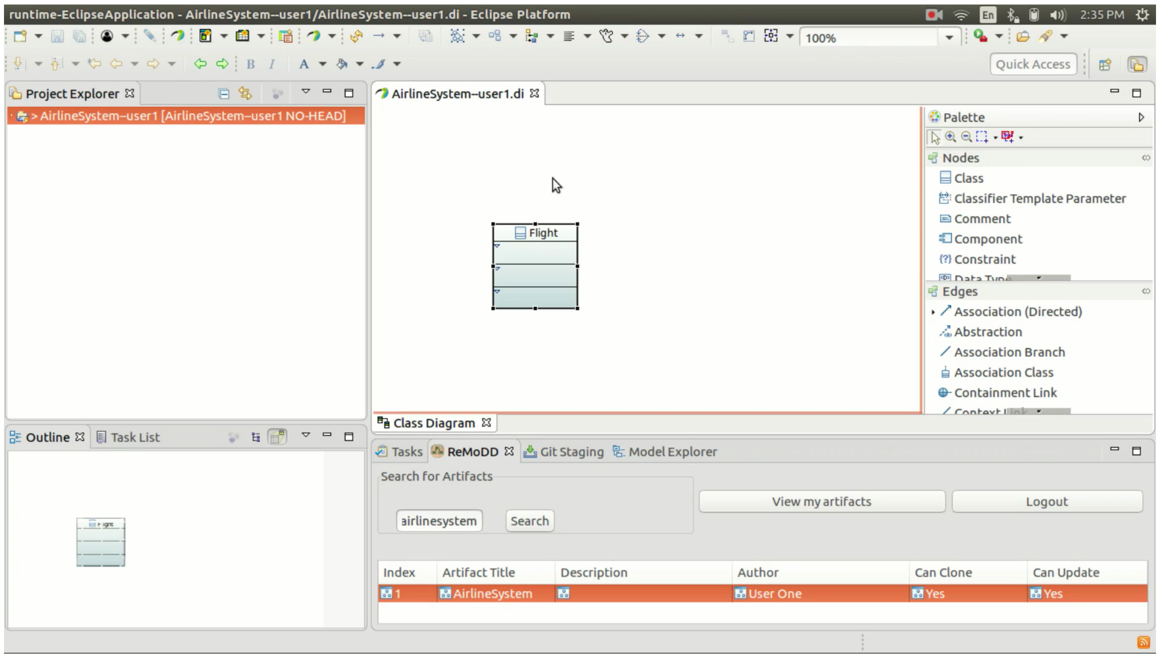
Artifact Types
<none> Add
Artifact types
Model Remove
Select all terms that describe the types of artifacts in the submission.

System/Software Domains
<none> Add
Artifact domains
System Software Remove
Select all software/system types that apply to this submission, if any.

Lifecycle Phases
<none> Add
All selections
Design › System Remove
Select all applicable development lifecycle phases, if any.

Modeling Languages/Notations
<none> Add
Artifact languages/notations
UML › class diagram Remove

Fig. 1. Scenario 1: Entering Metadata for Artifact Creation



Search for Artifacts
airlinesystem Search
View my artifacts Logout

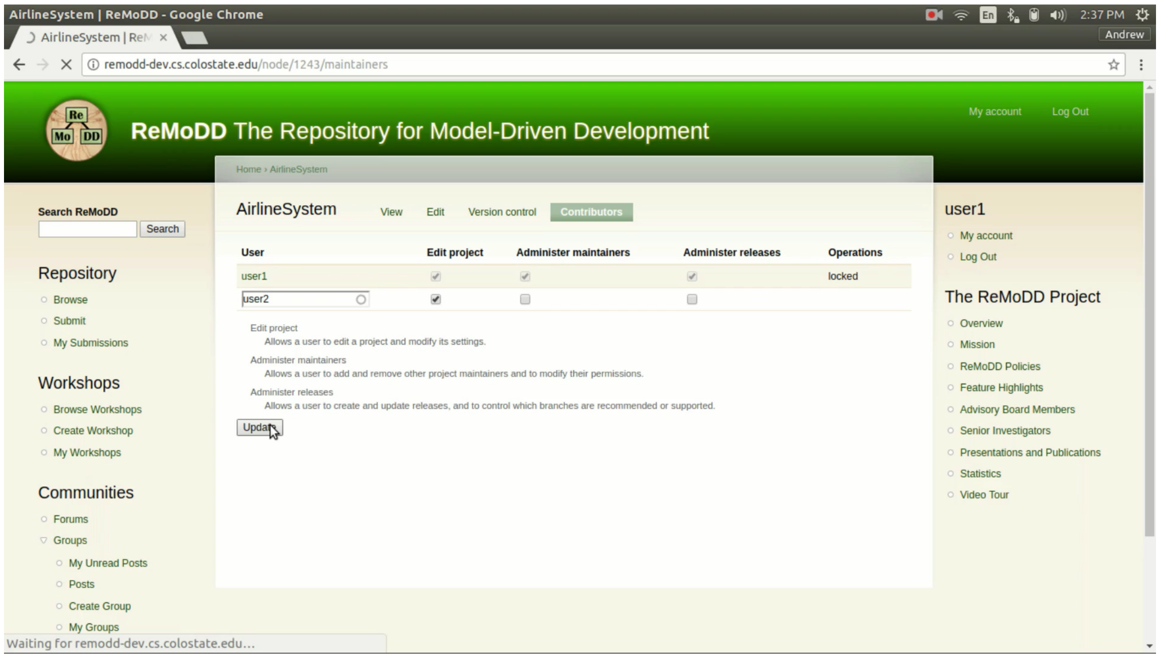
Index	Artifact Title	Description	Author	Can Clone	Can Update
1	AirlineSystem		User One	Yes	Yes

Fig. 2. Scenario 2: Creating Class Diagram Using Papyrus

Scenario-2: Next, we show how the user can access REMODD and its artifacts as an Eclipse plug-in. From Eclipse, the user opens a new window and selects the REMODD view. The user then logs in to REMODD using their REMODD credentials and performs searches similar to how it is done via the web-interface. The user can select the previously created artifact (i.e., AirlineSystem) for editing. Figure 2 shows how the user searches for the AirlineSystem project for cloning, and then edits the project to include a class diagram containing a single class (i.e., Flight), using the Papyrus modeling tool.

After completing the editing process, the user pushes the changes back to REMODD by (1) selecting the "Team" option from the Eclipse Project menu, (2) selecting the "Commit" option to bring up a window to enter a commit message, and (3) selecting the "Push and Commit" button to complete the process of submitting the changes to REMODD. At this point, the user clicks on the REMODD tab and logs out from the REMODD website.

Scenario-3: As with many MDE projects, multiple collaborators may work on a given artifact over a given time. This


AirlineSystem | ReMoDD - Google Chrome

remodd-dev.cs.colostate.edu/node/1243/maintainers

ReMoDD The Repository for Model-Driven Development

Home › AirlineSystem

AirlineSystem — View | Edit | Version control | Contributors

User	Edit project	Administer maintainers	Administer releases	Operations
user1	✓	✓	✓	locked
user2	✓	☐	☐	

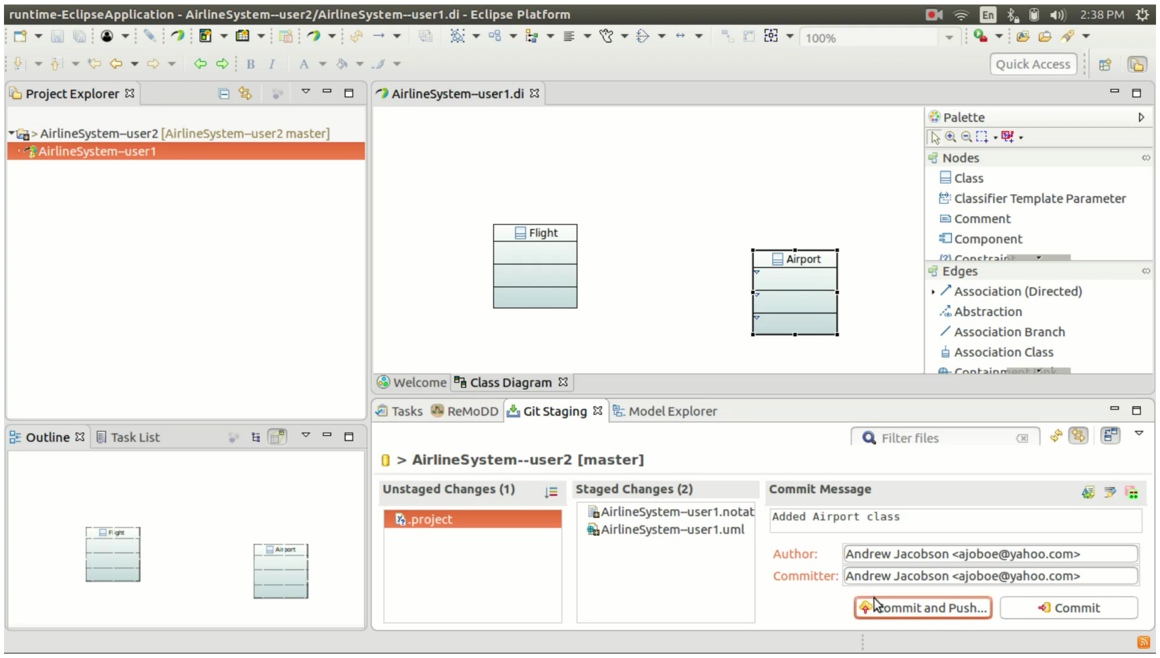
Edit project
Allows a user to edit a project and modify its settings.

Administer maintainers
Allows a user to add and remove other project maintainers and to modify their permissions.

Administer releases
Allows a user to create and update releases, and to control which branches are recommended or supported.

Update

Fig. 3. Scenario 3a: Adding Second User as Contributor to Project


runtime-EclipseApplication - AirlineSystem--user2/AirlineSystem--user1.di - Eclipse Platform

Project Explorer: > AirlineSystem--user2 [AirlineSystem--user2 master] / AirlineSystem--user1

AirlineSystem--user1.di: Flight, Airport

Git Staging: > AirlineSystem--user2 [master]

Unstaged Changes (1): .project

Staged Changes (2): AirlineSystem--user1.notat, AirlineSystem--user1.uml

Commit Message: Added Airport class

Author: Andrew Jacobson <ajoboe@yahoo.com>

Committer: Andrew Jacobson <ajoboe@yahoo.com>

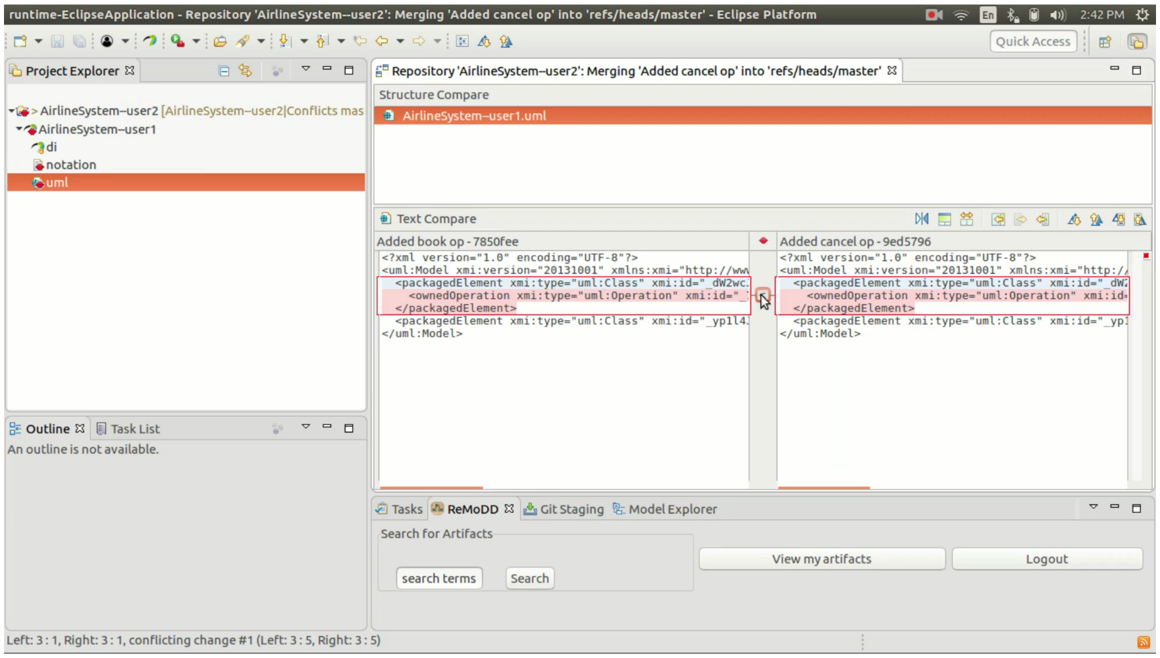
Commit and Push... | Commit

Fig. 4. Scenario 3b: Class Diagram Modified by Second User

scenario illustrates how REMODD supports the feature of adding and managing contributors for artifacts. The user starts with the Artifacts page for a given artifact, and select the "Contributors" tab. As shown in Figure 3, the user enters the name of a new contributor and the corresponding privileges ("Edit Project", "Administer Maintainers", or "Administer Releases"), and clicks the "Update" button at the bottom.

We next illustrate how the new contributor, *user2* opens the Eclipse project to access the AirlineSystem project, and then opens the class diagram to add a new class called Airport. Figure 4 shows the resulting class diagram. Upon saving the class diagram, *user2* repeats the process of saving the project and pushing the revised artifact to the REMODD repository.

Scenario-4: This scenario represents a commonly occurring situation where two or more modelers check out a version and perform conflicting changes. These conflicts must be resolved when the models are merged. This scenario starts


runtime-EclipseApplication - Repository 'AirlineSystem--user2': Merging 'Added cancel op' into 'refs/heads/master' - Eclipse Platform
Project Explorer
AirlineSystem--user2 [AirlineSystem--user2|Conflicts mas
AirlineSystem--user1
di
notation
uml
Repository 'AirlineSystem--user2': Merging 'Added cancel op' into 'refs/heads/master'
Structure Compare
AirlineSystem--user1.uml
Text Compare
Outline
Task List
An outline is not available.
Tasks
ReMoDD
Git Staging
Model Explorer
Search for Artifacts
search terms
Search
View my artifacts
Logout

Added book op - 7850fee

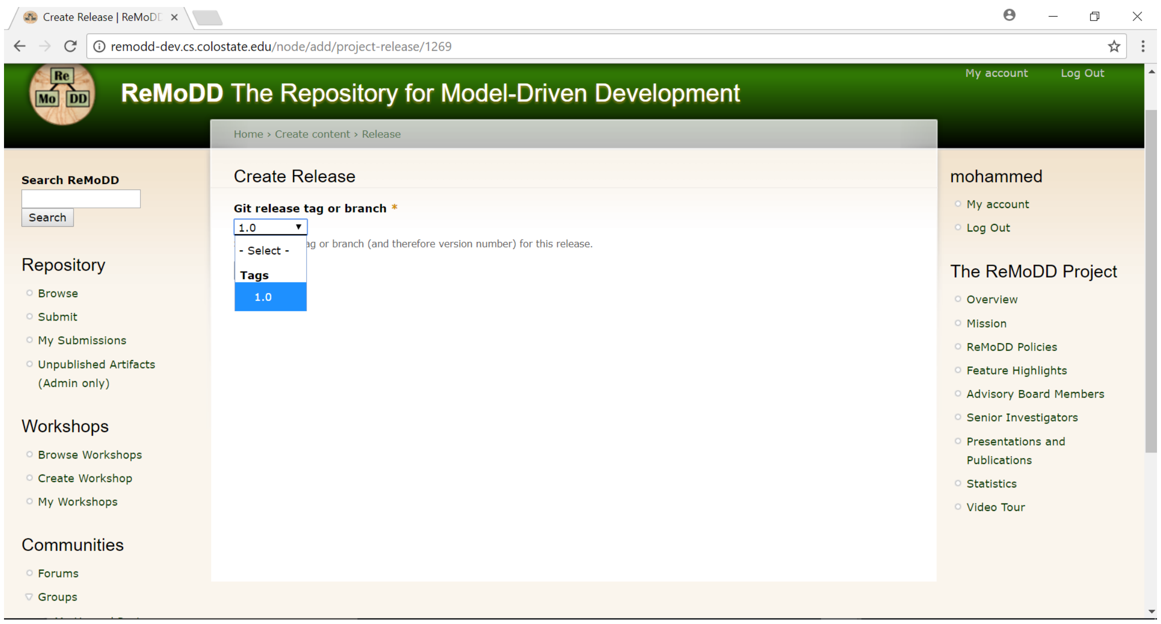
```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Model xmi:version="20131001" xmlns:xmi="http://www
  <packagedElement xmi:type="uml:Class" xmi:id="_dW2wc
    <ownedOperation xmi:type="uml:Operation" xmi:id="_
  </packagedElement>
  <packagedElement xmi:type="uml:Class" xmi:id="_yp1l4
</uml:Model>
```

Added cancel op - 9ed5796

```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Model xmi:version="20131001" xmlns:xmi="http://
  <packagedElement xmi:type="uml:Class" xmi:id="_dW
    <ownedOperation xmi:type="uml:Operation" xmi:id=
  </packagedElement>
  <packagedElement xmi:type="uml:Class" xmi:id="_yp
</uml:Model>
```

Left: 3 : 1, Right: 3 : 1, conflicting change #1 (Left: 3 : 5, Right: 3 : 5)

Fig. 5. Scenario 4: Merge Conflicts



Create Release

Git release tag or branch *

1.0

- Select -

Tags

1.0

ag or branch (and therefore version number) for this release.

Fig. 6. Scenario 5: Add Git Tag Release Number

after Scenario 3, when *user1*'s diagram only has the Flight class, and *user2*'s diagram has both Flight and Airport, which have been pushed to REMODD. First, *user2* adds an operation called `book()` to the Flight and commits the change but does not push. Next, *user1* pulls the changes that were pushed, and thus, gets the Airport but not the `book()` operation. Next, *user1* proceeds to add the `cancel()` operation to the Flight, and commits and pushes the change. When *user2* pulls changes from REMODD, there is a conflict because of the `book()` and `conflict()` operations. There is a built-in merge tool in e-git that shows the conflicts to be merged as shown in Figure 5.

Scenario-5: This scenario is needed only when a contributor wants to create a downloadable release of the artifact and make it available to all REMODD users on the website, such as after Scenarios 2, 3, and 4. We illustrate how this step is performed after Scenario 4. After *user2* pushes the revised artifact to REMODD, they create a `git` tag for the pushed commit. The tag number becomes the version number for the downloadable release. Next, *user2* opens the AirlineSystem artifact page from the REMODD website, clicks on the "Add new release" option, selects the tag number (see Figure 6), adds a short description of the release, and finally clicks on "Save". The created release contains the model files archived and attached. REMODD users can view and download this artifact release.

Scenario-6: This scenario shows how REMODD users who are not on the artifact's contributor list can still search for the artifact and download it via the plug-in so that they can edit and store versions locally without committing changes back to REMODD or creating new releases. This scenario is demonstrated with *user3*, who is not a contributor for the AirlineSystem artifact. *User3* logs into REMODD via the plug-in, searches for the AirlineSystem artifact, and then clones it to a project in Eclipse. At this point, *user3* is allowed to edit the artifact files in the project (not shown in the video).

III. Related Work

REMODD can potentially be used with other repositories and complementary modeling environments through its new API, a few of which we describe below. The Generic Modeling Environment (GME) [2] allows users to create domain specific modeling languages and code generation environments. GME provides a repository to store developed models in a database or in XML format. GenMyModel [3] is a web-based toolset that allows users to edit UML models in the cloud. It enables users to share models with other users or on social networks. The Model-Aware Repository and Service Environment (Morse) [4] provides a service-based repository for the storage and retrieval of models. Morse supports versioning capabilities. MDEForge [5] is an extensible web-based modeling framework that provides a community-based modeling repository. It supports mechanisms to find artifacts, and provides web access and other API-based services that enable the management of the artifacts, metamodels, and model transformations. The OCL repository[3] contains examples of Object Constraint Language expressions, where it is hosted by GitHub and allows users to contribute without having to register. OOModels[4] is an open library of object-oriented modeling. Users can use OOModels to download and discuss artifacts, develop modeling artifacts, and find software compatible with their models.

IV. Conclusions and Future Work

REMODD now includes an interface with `git` support and an API to enable the MDE community to integrate their development environments with REMODD. Users can download artifacts from the repository and create new versions in their respective development environments. An Eclipse plug-in uses the API to implement this functionality. We plan to further assess the usability and utility of these new features. We also plan to collaborate with developers of other modeling tools and repositories to enable them to interact with REMODD.

[3]https://github.com/jcabot/ocl-repository/

[4]http://oomodels.org/page/Main_Page/

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants No. CNS 1305381 and CNS 1305358.

References

[1] R. B. France, J. M. Bieman, S. P. Mandalaparty, B. H. C. Cheng, and A. C. Jensen, "Repository for model driven development (ReMoDD)," in *34th International Conference on Software Engineering, ICSE 2012, June 2-9, 2012, Zurich, Switzerland*, 2012, pp. 1471–1472.

[2] A. Ledeczi, M. Maroti, A. Bakay, G. Karsai, J. Garrett, C. Thomason, G. Nordstrom, J. Sprinkle, and P. Volgyesi, "The generic modeling environment," in *Workshop on Intelligent Signal Processing, Budapest, Hungary*, vol. 17, 2001, p. 1.

[3] M. Dirix, A. Muller, and V. Aranega, "Genmymodel: an online uml case tool," in *ECOOP*, 2013.

[4] T. Holmes, U. Zdun, and S. Dustdar, "Automating the management and versioning of service models at runtime to support service monitoring," in *Enterprise Distributed Object Computing Conference (EDOC), 2012 IEEE 16th International*. IEEE, 2012, pp. 211–218.

[5] F. Basciani, J. Di Rocco, D. Di Ruscio, A. Di Salle, L. Iovino, and A. Pierantonio, "Mdeforge: an extensible web-based modeling platform." in *CloudMDE@ MoDELS*, 2014, pp. 66–75.