A Dataset of EMF Models from Eclipse Projects

Stefan Kögel
stefan.koegel@uni-ulm.de

Matthias Tichy
matthias.tichy@uni-ulm.de

September 13, 2018

1 Overview

Models are key artefacts in Model-driven software engineering. Data sets of models from practice are highly valuable as input for different modelling research areas, e.g., performance benchmarks for modelling tools and analysing model transformations, as well as in empirical research, e.g., understanding how models are designed and evolve over time.

Unfortunately, there is a lack of data sets containing models, their meta models, and their evolution history. We present such a data set and describe our data collection method.

The Eclipse modeling framework (EMF) is the major framework for developing and using EMF models providing a rich ecosystem developing many models and meta models. Thus, we mined meta models and their instances from git repositories associated with Eclipse projects [1], including their version history.

Our data set was created on 2018-05-25 and contains 31799 models of which 4732 are meta models with a total of 101267 versions. These were mined from 247 repositories belonging to 130 projects hosted on Eclipse Projects. Our data set is made available as a `sqlite3` data base and the corresponding ER-diagram can be found in Figure 1. We also extracted all model versions as stand alone files in a directory structure (see Section 2.5 and Section 3).

The next section of this report describes what kinds of data we collected and our detailed data collection procedure. In the last section, we give an illustrative example to explain how to work with our data set.

[1] `https://www.eclipse.org/projects/` (accessed 2018-05-25)

2 Data Collection

Our data collection process consisted of three steps: (1) Find git repositories associated with Eclipse Projects via web scraping, (2) clone the found repositories, and (3) collect information about meta models, instance models, and their versions from the repositories.

To provide further details of our data collection process we included the scripts that we used to create the data set together with the data set. For further questions contact `stefan.koegel@uni-ulm.de` .

2.1 Projects and Repositories

At first, we crawled the eclipse projects website [2] and gathered project names, URLs and associated repositories. We found associated repositories by searching for `span` tags with the class `.clone-repo-url` whose content matched the regular expression

```
^(https?|git)://(github.com/eclipse|git.eclipse.org)(.*/)([^/]+)$
```

For repositories, we saved their URLs and extracted their names from these URLs. We also saved the number of commits from the repositories after cloning them (see Section 2.2). The resulting data is represented in the tables `repositories`, `repositories_to_projects`, and `projects`.

While crawling, we obeyed the `robots.txt` and only requested one page per ten seconds. We used the Scalpel [3] web scraper library for Haskell.

2.2 Commits

After collecting the repository URLs, we cloned them. This took several hours and resulted in downloading about 60GB of data.

We used the following git command to extract all commits from a repository in chronological order

```
git log --all --format=%H --date-order --reverse
```

then we used the commands

```
git show -s --format=[%s|%b|%ct] <commit_hash>
```

to extract the subject (`%s`), body (`%b`), and commit date (`%ct`) from each commit. We did not store the author name to keep our data set anonymous.

Out of the 906 repositories that we cloned, we could only find models in 247 (28%) of them (see Section 2.3 for more details). We kept the repositories without models in our data set for statistical purposes, but did not include their commits.

[2] `https://projects.eclipse.org/` (accessed 2018-05-25)
[3] `https://hackage.haskell.org/package/scalpel` (accessed 2018-05-25)

2.3 Artefacts and their Versions

The Eclipse Modeling Framework has two types of models: meta models and instance models. The structure of an instance model is defined by its meta model. In order to collect all versions of all models, we performed two passes, first collecting all versions of meta models and then collecting all versions of instance models.

A single model can have many different versions whose file paths or file names may change. Thus, we describe a model as an abstract artefact with a unique id that has multiple versions. We also store the number of versions a specific artefact has and whether it is a meta model.

In our first pass, we identified meta models in the cloned repositories by searching for files with the `.ecore` extension. Our second pass is similar to the first one. However, we cannot simply search for files with a specific extension as before as instance models can have arbitrary file extensions. Instead, we exploit the fact that all serialized instance models reference the `xmi` namespace. Hence, we checked for all files whether they conform to XML and if yes, checked whether they contained the `xmi` namespace using an XML parser [4] .

For every model that we found, we recorded which commits changed it. For every such change, we recorded the file path of the model after the change and the number of added and deleted lines. A version can be uniquely identified by its file path together with its commit hash. We also numbered the versions of an artefact in chronological order.

The command

`git log --follow -M50% --format=%H <filepath>`

produces a list of all hashes of commits (`%H`) that changed the model. The command

`git log --follow -M50% --numstat --ignore-space-change --ignore-blank-lines --format= <filepath>`

produces a list of added and deleted lines (`--numstat`). And the command

`git log --follow -M50% --name-only --format= <filepath>`

produces a list of the model versions' filepaths (`--name-only`).

Unfortunately, git does not track when files are renamed, not even when using the `git mv` command. The above git commands use the arguments `--follow -M50%` to specify, that if a commit deletes a file and creates a new one and these files share at least 50% of their content, then git will treat this as a renaming of the file. This allows us to follow the version history of a model even if it was renamed. We used 50%, because it is the default value used by git.

Using the command

`git show <commit_hash>:<file_path>`

[4] xml-conduit `http://hackage.haskell.org/package/xml-conduit` (accessed 2018-05-25)

we extracted all versions of meta and instance models from the git repositories, parsed them using an XML parser, and recorded the namespaces used by them. We also computed the `sha1` hashes of the files' contents and included them in our data set.

2.4 Namespaces

Meta models generally define a namespace that is then referenced by their instance models. We used the namespace data from Section 2.3 to reconstructed the reference graph between the instance models and their meta models.

2.5 Contents of all Versions

We have extracted textual contents of all model versions into directories, according to the following schema:

`models/`*`<project_id>`*`/`*`<artefact_id>`*`/`*`<version_id>`*`.model`

3 Usage Example of our Data Set

The simplest way to interact with our data set is to use a `sqlite3` [5] console.

In the following example, our aim is to find the five models (meta & instance) with the highest number of added and deleted lines over all versions that have at least three versions:

```
SELECT repository_id, artefact_id,
       sum(additions) + sum(deletions) AS line_changes
FROM repositories NATURAL JOIN artefacts NATURAL JOIN versions
GROUP BY artefact_id
HAVING count(*) > 3
ORDER BY line_changes DESC
LIMIT 5;
```

The result of the above query is:

```
repository_id | artefact_id | line_changes
--------------+-------------+-------------
91            | 9178        | 602805
28            | 1174        | 509745
91            | 9177        | 480493
65            | 2404        | 226907
28            | 1192        | 217998
```

Thus, the directory `models/91/9178` (see Section 2.5) contains the contents of the versions of the first model from the above query result: `34921.model, 34922.model, 34923.model, 34924.model, 34925.model` for further detailed processing.

[5] `https://www.sqlite.org/index.html`

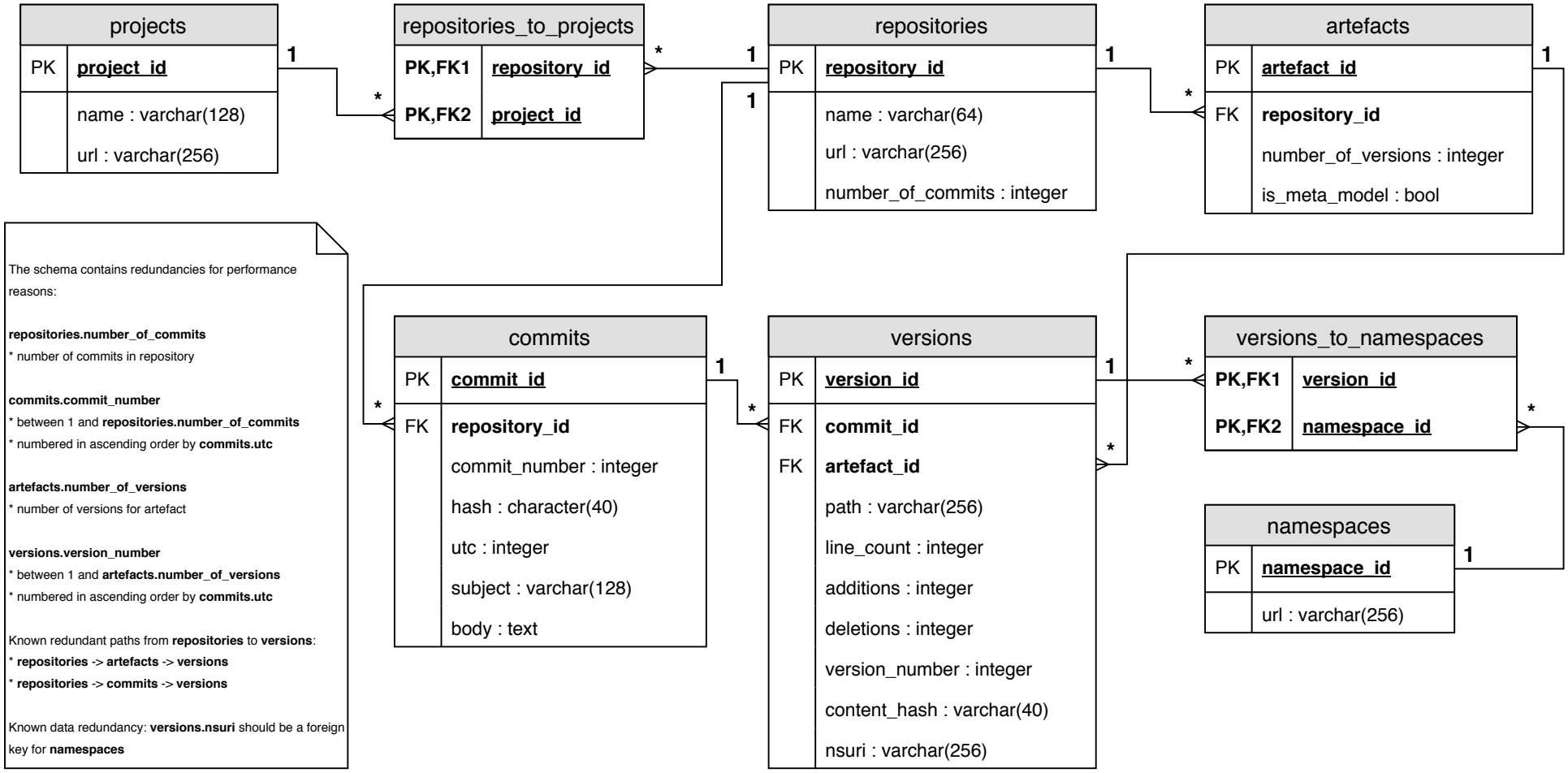

Figure 1: ER-Diagram of `sqlite3` data base.