# Sample Use-case: University Management System

Student: Hoang Nguyen Phuoc Bao

January 11, 2021

## Requirements

Before going through this sample use-case, please make sure that you have the Eclipse installation with all required plugins on your running computer:[1] As a suggestion, you can have the required installation of Eclipse via the following options:

- For one who does not have Eclipse installed, please download the following complete installation, with all plugins used in the use case via this link:

  `http://miso.es/teaching/mde1718/eclipse.rar`

- For one who does have Eclipse installed, please install the following plug-ins:
  - Work with: 2020-09 - http://download.eclipse.org/releases/2020-09
    - General Purpose Tool
      - Eclipse Plug-in Development Environment
    - Modeling
      - Acceleo
      - ATL SDK - ATL Transformation Language SDK
      - Ecore Diagram Editor (SDK)
      - EMF - Eclipse Modeling Framework SDK
      - OCL Examples and Editors SDK
      - Xtext Complete SDK
  - Work with: http://download.eclipse.org/modeling/gmp/gmf-tooling/-updates/releases-3.2.1/
  - Work with: http://download.eclipse.org/modeling/emft/henshin/updates/release

In addition, Java Version 8 or above is also a requirement.

[1] Taken from the content of the Formal Model Driven Engineering class.

## Execution Instructions

### Step 1: Import all projects

Note: In case you clone the project from GitHub, please make sure to clone the branch:
`university-use-case`.

```
git clone https://github.com/npbhoang/MDS-SQLSI.git
cd MDS-SQLSI
git checkout university-use-case
```

1. Run Eclipse.
2. Import the related projects:
   - On the tool bar: `File / Import`
   - In the `Import` pop-up window:`General / Existing Projects into Workspace`
   - Navigate correctly the downloaded project.
   - Add all projects.

Figure 1 shows what you should obtain in your `Package Explorer` afterwards.

### Step 2: Register EPackages and Open Runtime-Eclipse

Register EPackages:

1. Navigate to `uam.mde20.sqlsi.datamodels / metamodels / datamodels`
2. Right click and choose `Register EPackages`
3. Do the same for project `uam.mde20.sqlsi.securitymodels` and `uam.mde20.sqlsi.relationaldbmodels`

Open Runtime-Eclipse:

1. Navigate to `uam.mde20.sqlsi.xtext.securitymodels`.
2. Right click and choose `Run as / Eclipse Application / Launch Runtime Eclipse` (Choose the second one).
3. Please ignore the error warning.
4. In the new Eclipse instance, import project just like Step 1, but this time, only import `usecase.university` project.

Figure 2 shows what you should obtain in your `Project Explorer` in the new Eclipse instance afterwards.

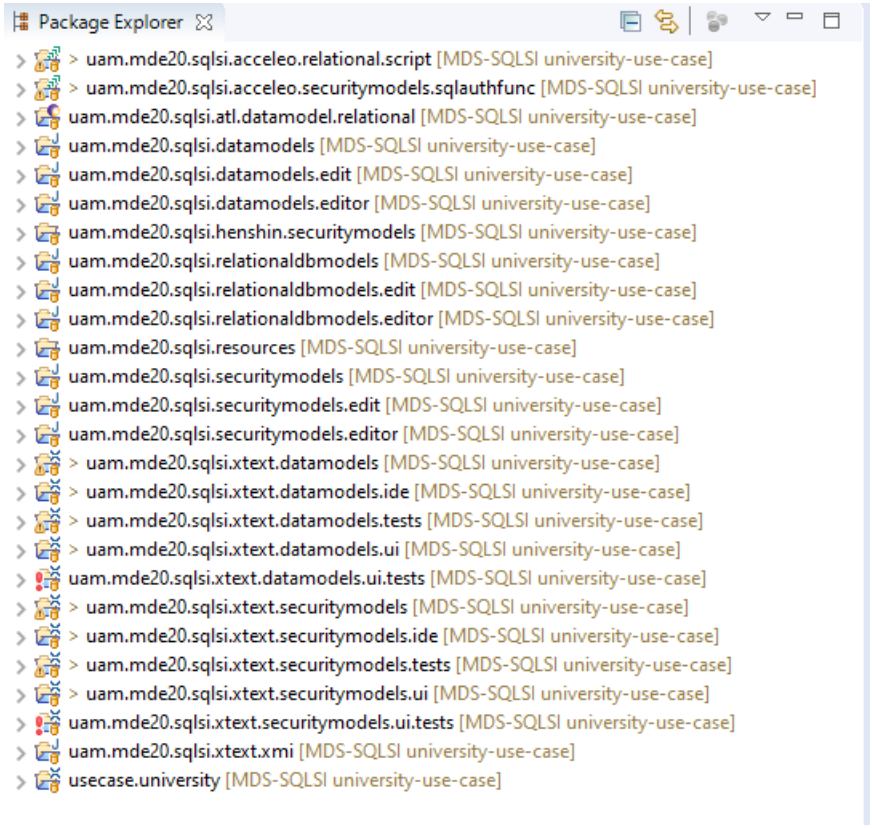


Figure 1: Package Explorer after Step 1

**Step 3: Generate models from specific DSL format to XMI format**

1. Switch back to the original Eclipse window.
2. Navigate to `uam.mde20.sqlsi.xtext.xmi / src / xmi / Main.java`.
3. Please change the absolute path of the DSL model of datamodel (`universityDM.dm` in the `usecase.university` project) and securitymodel (`universitySM.sm` in the `usecase.university` project) accordingly.
4. Right click on `Main.java` and choose `Run as / Java Application`.
5. Refresh the `usecase.university` on both Eclipse windows.

Figure 3 shows what you should obtain in your `Project Explorer` in the new Eclipse instance afterwards.

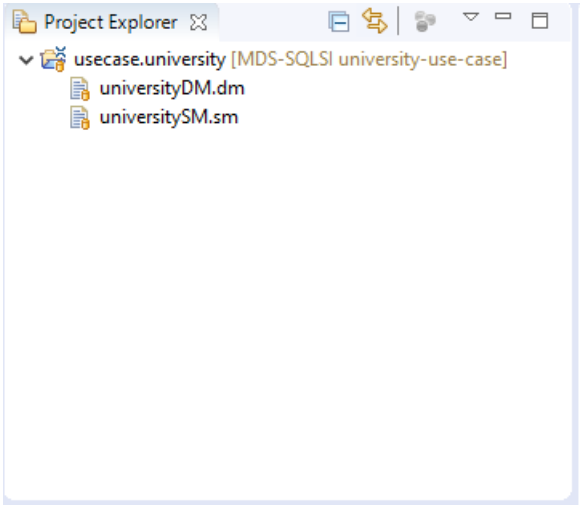


Figure 2: Project Explorer of new Eclipse instance after Step 2

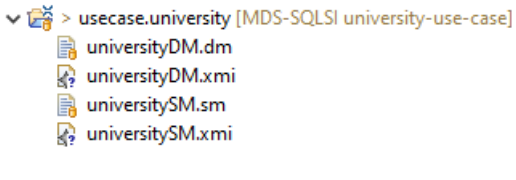


Figure 3: Project Explorer of new Eclipse instance after Step 3

**Step 4: Transform datamodel to relational database model from which generate SQL schemata**

Transform datamodel to relational database model:

1. Switch back to the original Eclipse window.
2. On the toolbar: `Run / Run Configurations...`
3. Choose `ATL Transformation`
4. Choose `ATL Module` as `/uam.mde20.sqlsi.atl.datamodel.relational/transformations/dm2relational.atl`
5. Figure 4 shows a sample configuration of this transformation.
6. Click `Run`
7. Refresh the `usecase.university` on both Eclipse windows.

Generate SQL schemata from relational database model:

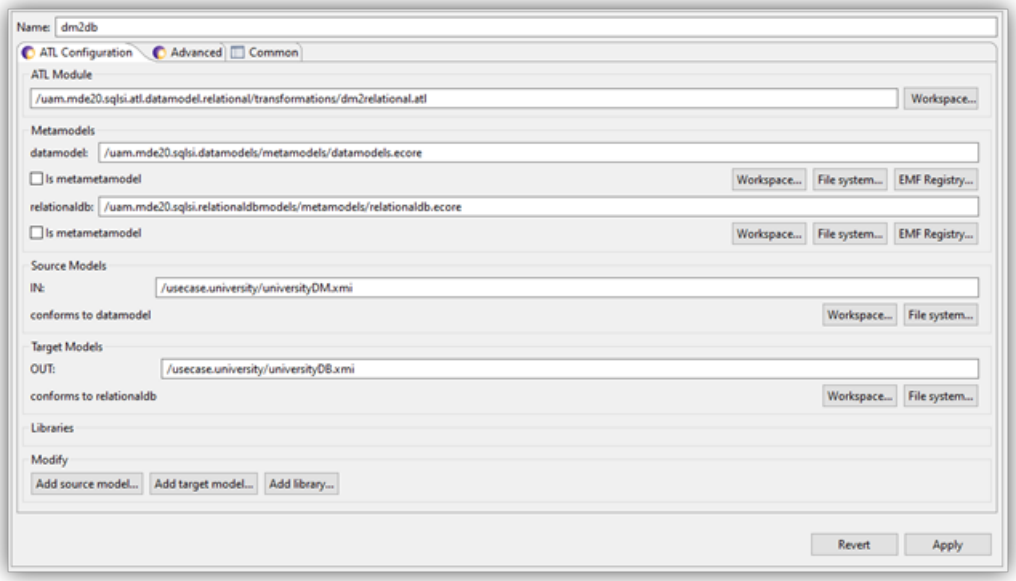


Figure 4: ATL Transformation sample configuration

1. On the toolbar: `Run / Run Configurations...`
2. Choose `Acceleo Application`
3. Figure 5 shows a sample configuration of this generation.
4. Click `Run`
5. Refresh the `usecase.university` on both Eclipse windows.

Figure 6 shows what you should obtain in your `Project Explorer` in the new Eclipse instance afterwards.

### Step 5: Manipulate security model then generate SQL authorization functions

Manipulate security model:

1. Switch back to the original Eclipse window.
2. Navigate to `uam.mde20.sqlsi.henshin.securitymodels / default.henshin`
3. Right click and choose `Henshin / Apply Transformation`
4. Figure 7 shows a sample configuration of this manipulation.

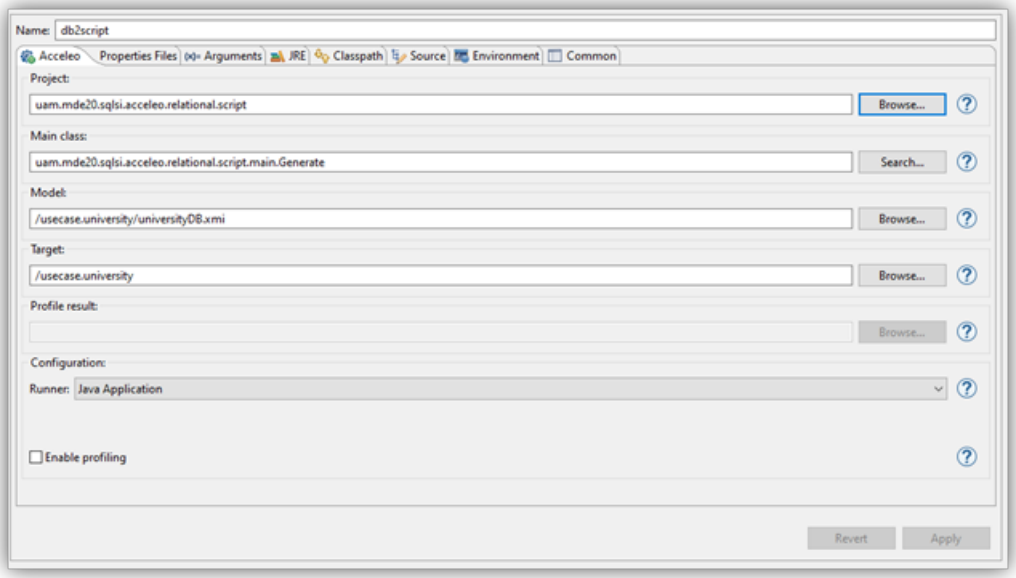


Figure 5: Acceleo Code-Generation sample configuration

5. Click `Transform`, the following error in Figure 8 window will pop-up. This is indeed a unsolved problem. Please, for the moment, ignore this error by click `OK`.
6. Click `Transform` again, this time it should work!
7. Refresh the `usecase.university` on both Eclipse windows.

Generate SQL authorization functions from security model:

1. On the toolbar: `Run / Run Configurations...`
2. Choose `Acceleo Application`
3. Figure 9 shows a sample configuration of this generation.
4. Click `Run`
5. Refresh the `usecase.university` on both Eclipse windows.

Figure 10 shows what you should obtain in your `Project Explorer` in the new Eclipse instance afterwards.

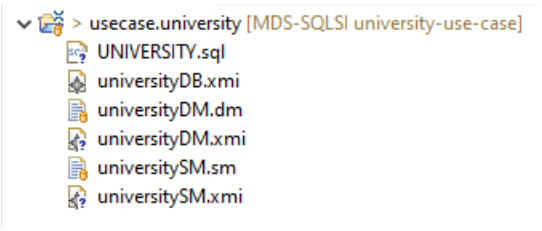


Figure 6: Project Explorer of new Eclipse instance after Step 4

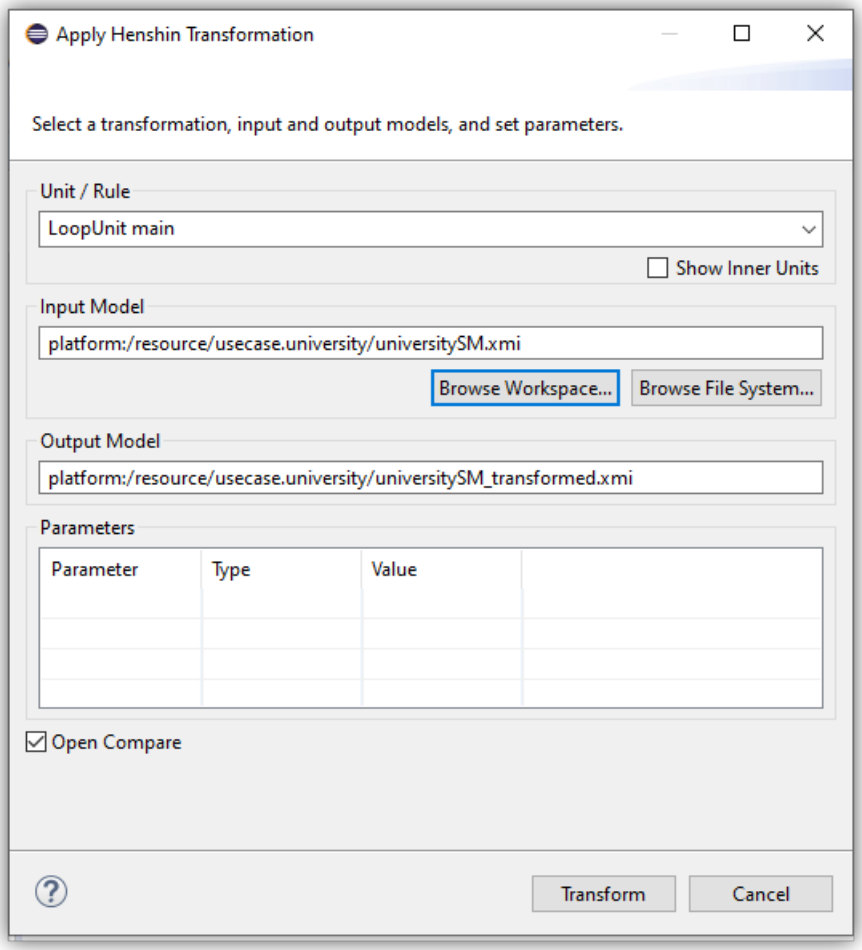


Figure 7: Henshin Transformation sample configuration

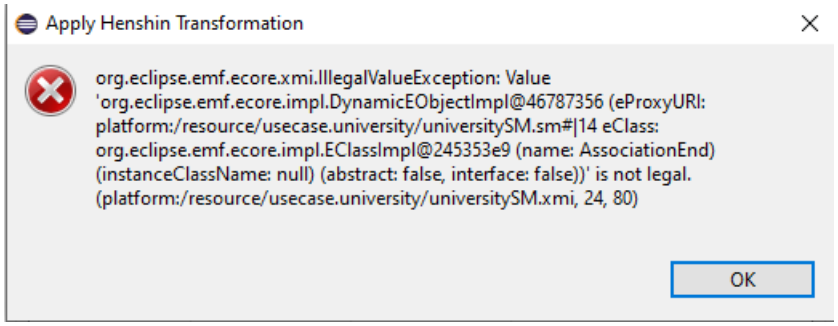


Figure 8: Henshin Transformation Error

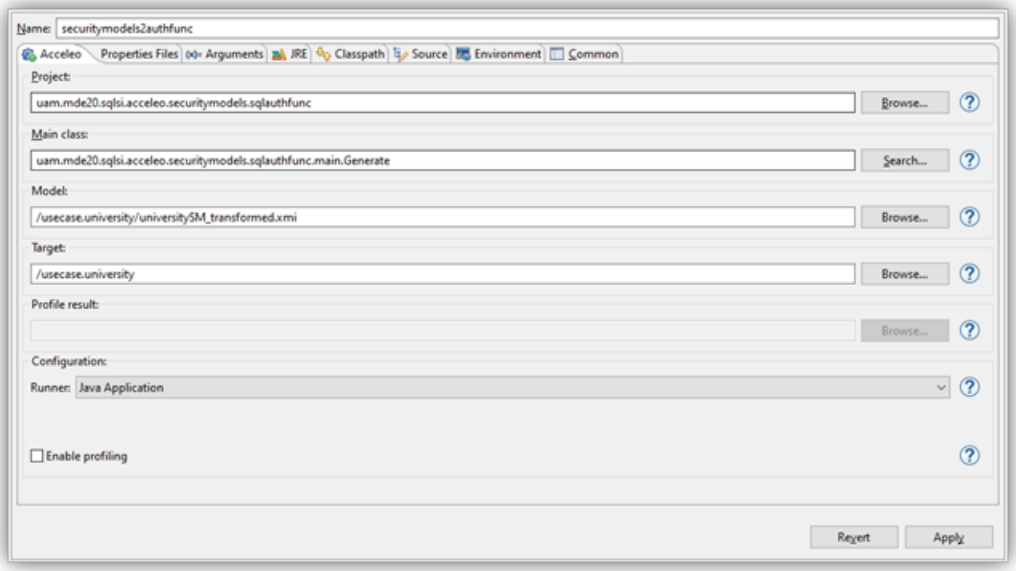


Figure 9: Acceleo Code-Generation sample configuration

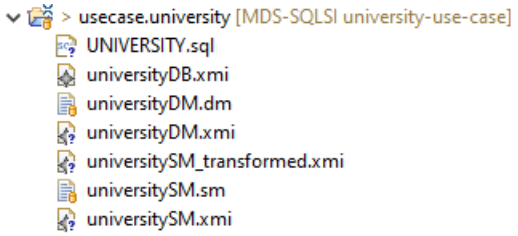


Figure 10: Project Explorer of new Eclipse instance after Step 5