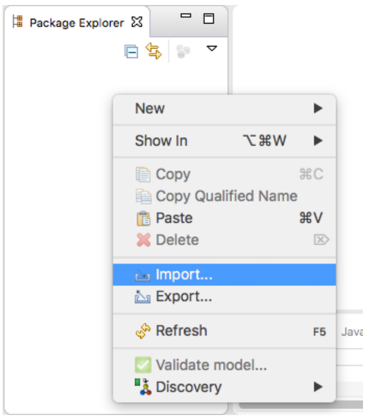
How to install and use SDG

This document: (1) presents how to you over the installation steps of the SDG tool (http://people.svv.lu/tools/SDG/), and (2) illustrates how to use SDG.

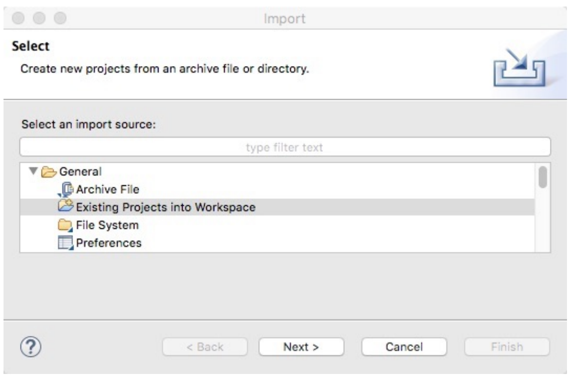
1- Installation steps

First, you need to install your Eclipse modeling environment and make sure that all system requirements are meet (see Section *System Requirements* of the tool webpage).

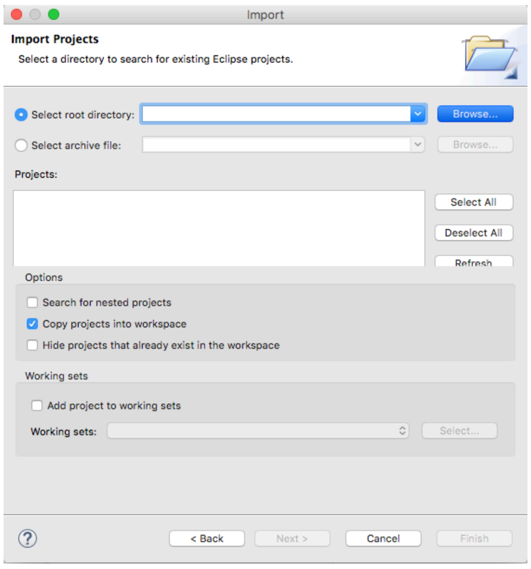
Then, download the SDG source code (see Section *Installation Material for SDG*). Once unzipped, import SDG into your Eclipse environment. Below, we illustrate how to import SDG.



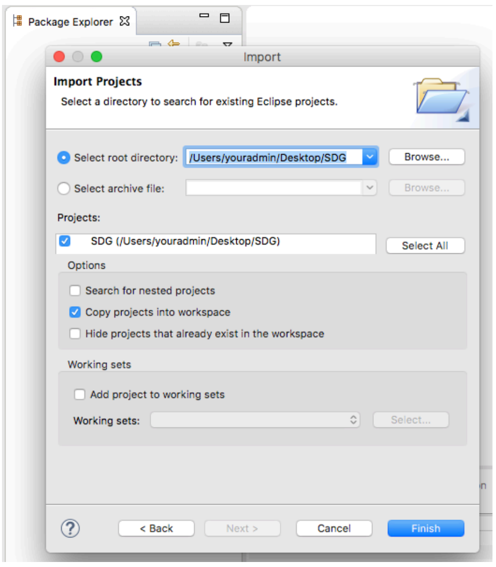
1



2

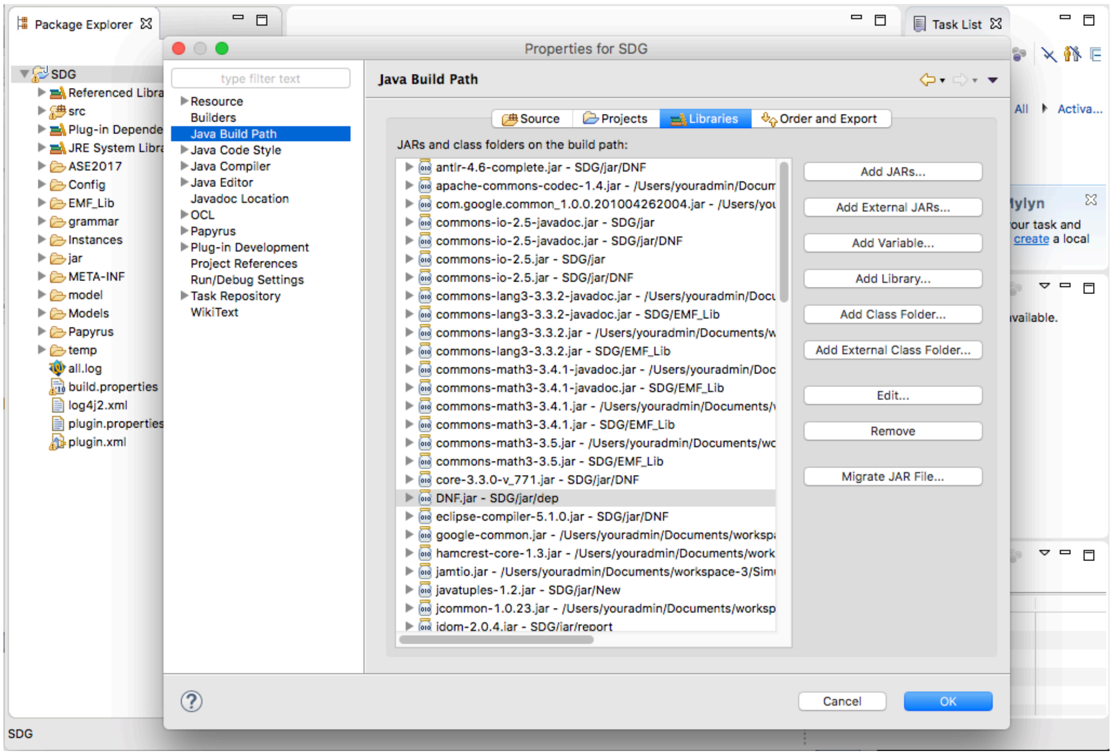

Import
Import Projects
Select a directory to search for existing Eclipse projects.
Select root directory:
Browse...
Select archive file:
Browse...
Projects:
Select All
Deselect All
Refresh
Options
Search for nested projects
Copy projects into workspace
Hide projects that already exist in the workspace
Working sets
Add project to working sets
Working sets:
Select...
< Back
Next >
Cancel
Finish

3


Package Explorer
Import
Import Projects
Select a directory to search for existing Eclipse projects.
Select root directory: /Users/youradmin/Desktop/SDG
Browse...
Select archive file:
Browse...
Projects:
SDG (/Users/youradmin/Desktop/SDG)
Select All
Options
Search for nested projects
Copy projects into workspace
Hide projects that already exist in the workspace
Working sets
Add project to working sets
Working sets:
Select...
< Back
Next >
Cancel
Finish

4

Next, make sure that all the third-party libraries under the *jar* and *EMF_Lib* folders in SDG are included in the project properties.



Java Build Path

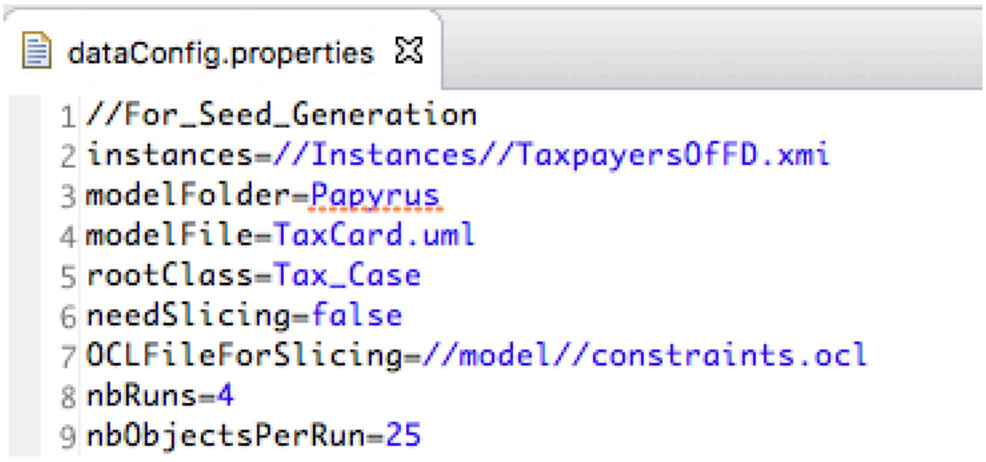
JARs and class folders on the build path:

```
antlr-4.6-complete.jar - SDG/jar/DNF
apache-commons-codec-1.4.jar - /Users/youradmin/Docum
com.google.common_1.0.0.201004262004.jar - /Users/you
commons-io-2.5-javadoc.jar - SDG/jar
commons-io-2.5-javadoc.jar - SDG/jar/DNF
commons-io-2.5.jar - SDG/jar
commons-io-2.5.jar - SDG/jar/DNF
commons-lang3-3.3.2-javadoc.jar - /Users/youradmin/Docu
commons-lang3-3.3.2-javadoc.jar - SDG/EMF_Lib
commons-lang3-3.3.2.jar - /Users/youradmin/Documents/w
commons-lang3-3.3.2.jar - SDG/EMF_Lib
commons-math3-3.4.1-javadoc.jar - /Users/youradmin/Doc
commons-math3-3.4.1-javadoc.jar - SDG/EMF_Lib
commons-math3-3.4.1.jar - /Users/youradmin/Documents/
commons-math3-3.4.1.jar - SDG/EMF_Lib
commons-math3-3.5.jar - /Users/youradmin/Documents/wc
commons-math3-3.5.jar - SDG/EMF_Lib
core-3.3.0-v_771.jar - SDG/jar/DNF
DNF.jar - SDG/jar/dep
eclipse-compiler-5.1.0.jar - SDG/jar/DNF
google-common.jar - /Users/youradmin/Documents/workspa
hamcrest-core-1.3.jar - /Users/youradmin/Documents/work
jamtio.jar - /Users/youradmin/Documents/workspace-3/Sim
javatuples-1.2.jar - SDG/jar/New
jcommon-1.0.23.jar - /Users/youradmin/Documents/worksp
jdom-2.0.4.jar - SDG/jar/report
```

2- Using the tool

a- Generating the seed data

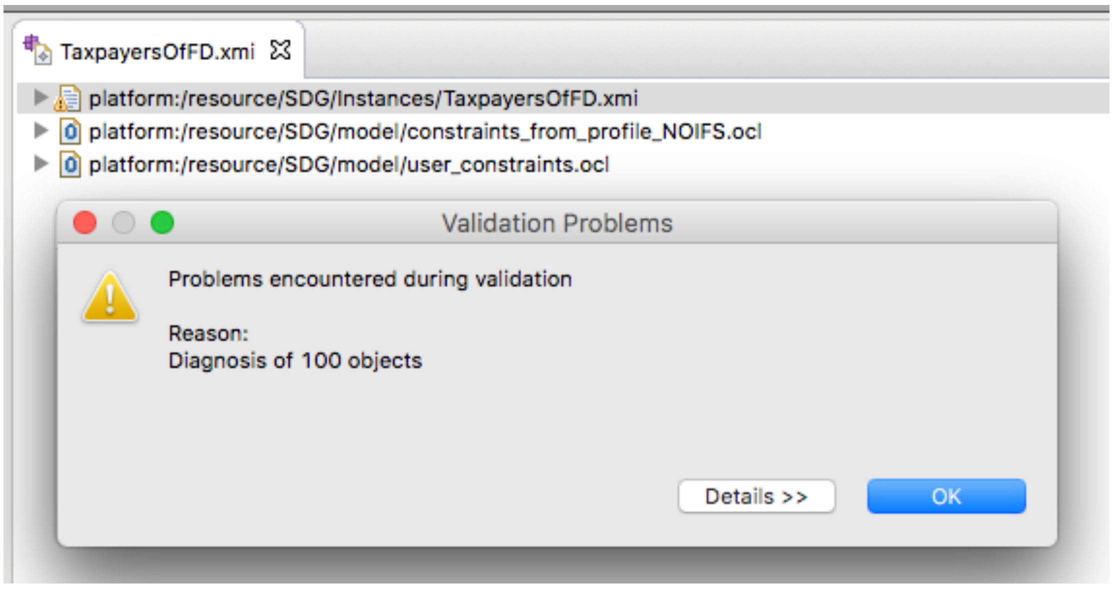
Run Main/GenerateSeeds to create the initial seed data from a given class diagram. By default, the data will be generated under the folder named *Instances* (file TaxpayersOfFD). Nevertheless, the output location and other relevant parameters can be parameterized through the file dataConfig.*properties* (under the folder *Config*) as shown bellow.



```
//For_Seed_Generation
instances=//Instances//TaxpayersOfFD.xmi
modelFolder=Papyrus
modelFile=TaxCard.uml
rootClass=Tax_Case
needSlicing=false
OCLFileForSlicing=//model//constraints.ocl
nbRuns=4
nbObjectsPerRun=25
```

- The parameter *instance defines* the path where the tool will store the seed data.
- The parameter *modelFolder* defines the path for the folder containing the domain model to instantiate.
- The parameter *modelFile* defines the file of the input class diagram (.uml).
- The parameter *rootClass* defines the name of the UML class that represents a single test case. In the figure above, our test cases are tax cases.
- The parameter *needSlicing* indicates whether data generation should cover all the elements of the input class diagram. If this parameter is set to true, then the seed generator will exclusively instantiate the UML elements provided as input via the parameter *OCLFileForSlicing* (a file containing OCL expression that refer to the elements to instantiate). Otherwise, that is when parameter *needSlicing* is set to false, data generation will cover all the classes, attributes and associations of the input class diagram.
- The parameters *nbRuns* and *nbObjectsPerRun* dfines how many instance models the seed generator should produce. In the figure above, the users want to generate up to 100 tax cases. We use two parameters instead of a single one, to optimize memory consumption and garbage collection processes.

Once seed generation complete, one can verify whether the seed data is logically valid against a set of OCL constraints. The constraints files are under the folder model (constraints_from_profile_NOIFS.ocl and user_constraints.ocl). As the seed generator is based on heuristics, often the data will be invalid as shown below:



platform:/resource/SDG/Instances/TaxpayersOfFD.xmi
platform:/resource/SDG/model/constraints_from_profile_NOIFS.ocl
platform:/resource/SDG/model/user_constraints.ocl

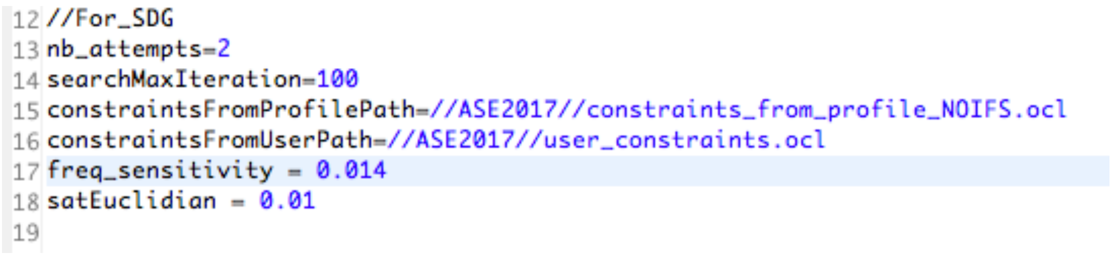
Validation Problems

Problems encountered during validation

Reason:
Diagnosis of 100 objects

b- Generating valid and representative data (running the core SDG)

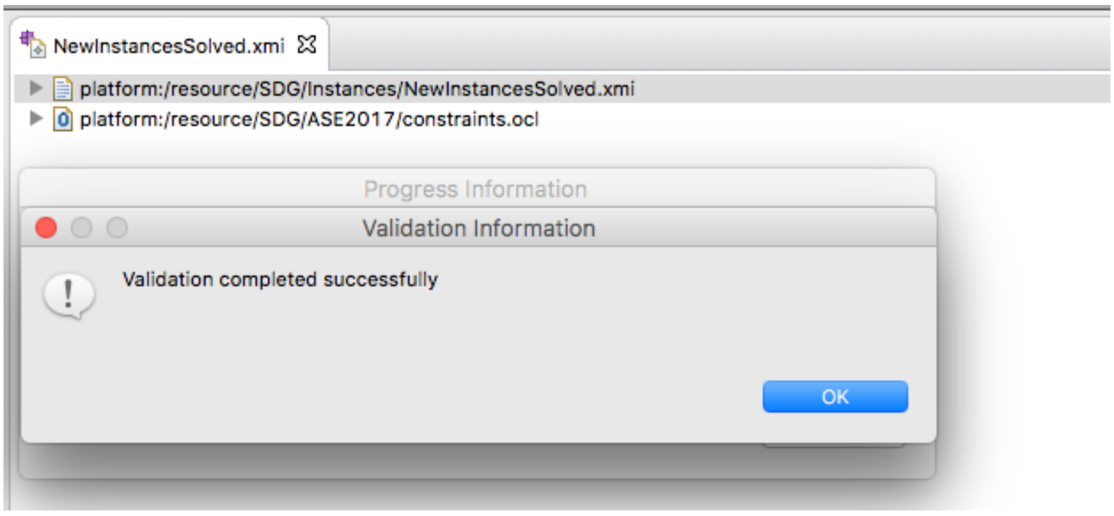
To generate valid and representative data from the seed data, run the SDG class that is inside the SDG's *Main* package. You can also customize SDG's parameters using the *dataConfig.properties* under the folder *Config*.



```
//For_SDG
nb_attempts=2
searchMaxIteration=100
constraintsFromProfilePath=//ASE2017//constraints_from_profile_NOIFS.ocl
constraintsFromUserPath=//ASE2017//user_constraints.ocl
freq_sensitivity = 0.014
satEuclidian = 0.01
```

- The parameter *nb_attempts* denotes the number of times that the OCL solver will be invoked over a given instance model to create tweaked instance models.
- The parameter *searchMaxIteration* defines the maximum number of iteration for the search employed by the solver.
- *constraintsFromProfilePath* and *constraintsFromUserPath* point to the OCL files containing the logical constraints to enforce over the data (either extracted from the class model itself or provided by users).
- The parameter *freq_sensitivity* denotes the margin beyond which two relative frequencies are deemed far apart.
- The parameter *satEuclidian* denotes the Euclidian distance threshold below which the data sample is considered as representative.

The final data sample is stored under the *instances* folder (*NewInstancesSolved.xmi*). One can now re-check the logical validity of the final data.



NewInstancesSolved.xmi

platform:/resource/SDG/Instances/NewInstancesSolved.xmi

platform:/resource/SDG/ASE2017/constraints.ocl

Progress Information

Validation Information

Validation completed successfully