**Pruning and Refactoring Tool User Manual**

# Overview

The Pruning and Refactoring tool (PR Tool) is for comparing a target UML information model with the source UML information model and recording the differences with Pruning and Refactoring Realizations.

# Functional Requirements

The functional requirements of the PR tool include IM clone, IM comparison, class copy, class split, class join, etc. The functional requirement documents are being developed.

# Programing Language

Programming language：JavaScript

Running environment: node.js

# History

|  |  |
| --- | --- |
| **Date** | **Features Added/Issues Fixed** |
|  |  |
|  |  |
|  |  |

# Implemented Functions

* 1. Model Comparison

Each pruning and refactoring association links the peer components between the source model and the target model.

1. Classes comparison

The features (including <<OpenModelClass>>) of peer classes compared between the source model and the target model include the following,

* The name of the class, shown as "name" in the comments,
* "support",
* "condition",
* "status",

"visibility",

* "isAbstract",
* "isActive",
* "isLeaf"

1. Attributes comparison

The features (including <<OpenModelAttribute>>) of peer attributes between the source model and the target model include the following,

* "name",
* "isReadOnly",
* "defaultValue",
* "isUses",
* "status"(LifeCycle),
* "isAbstract",
* "rpcType",
* "valueRange",
* "units",
* "path",
* "support",
* "condition",
* "isleafRef",
* "isOrdered",
* "isStatic",
* "isUnique",
* "aggregation",
* "visibility",
* "min-elements",
* "max-elements",
* "partOfObjectKey"

1. Associations comparison

The features of peer associations compared between the source model and the target model include the following,

* "name",
* "memberEnd1",
* "memberEnd2",
* "associationType",
* "type",
* "ownedEndName",
* "upperValue",
* "lowerValue"
  1. Class Copy

The user can specify the name of the class and the number of copies needed in “CopyAndSplit.txt”. After running the tool with the given information, the tool can provide the copied class in target model and update the PR realizations in mapping model. The copied class will have all the attributes in the identified copy class.

* 1. Class Split

The user can specify the name of the class and the number of splits needed in “CopyAndSplit.txt”. After running the tool with the given information, the tool can provide the splited class in target model and update the PR realizations in mapping model. The splited class will have no attributes.

* 1. Class Merge

The tool currently supports merging classes across multiple files or within one file. The user can specify which classes and attributes to merge by adding the <merge> stereotype in Papyrus. Before running the tool, the user need to import the following P&R profiles into Papyrus:



# How to Run This Tool

Running the Pruning and Refactoring tool takes the following steps.

* Step 1: Download nodejs from <https://nodejs.org/en/> , choose the latest version for your system.
* Step 2: Install nodesjs by double click the "node-vxxx.msi" file. Use the default options are fine. To test whether nodejs is properly installed, the user can type “node -v” in the terminal window. If the version is printed on the screen as shown in the picture below, you are good to go.



* Step 3: Clone the code from Github to your local repository. <https://github.com/HuiDingCATR/Pruning-and-Refactoring-Tool>
* Step 4: Copy the UML information model that you want to prune and refactor into the “project” folder. And rename it to “source.uml”.
* Step 5: In the command line, switch to the directory you put the PR tool. Type the following command to run the tool.

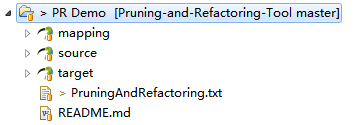
*node main.js*

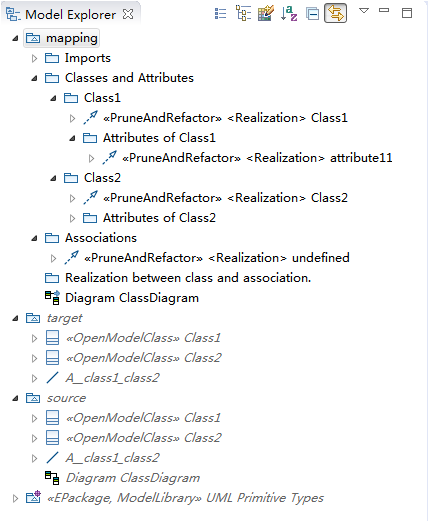
* Step 6: Two files will be generated in the project folder, target.uml and mapping.uml. If the user runs the tool for the first time, the target model will be the clone of the source model and the mapping model includes all the PR realizations between the two and the comparison details will be recorded in the comment of each realization. If the user runs the tool after updating the target model, the mapping model will be updated to reflect the differences.
* Step 7: If the user wants to copy a class, he/she can put the name of class and the number of copies needed in the CopyAndSplit.txt file. Note that, if no class copy function is needed, the file should not be present in the project folder.
* Step 8: After the CopyAndSplit.txt is properly filled, the user should run the tool again.

# View the Model in Papyrus

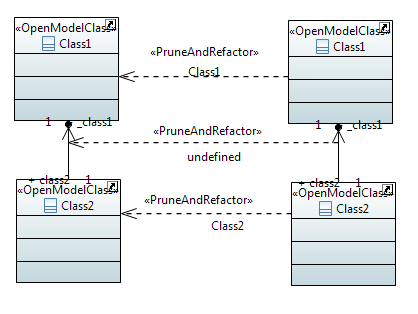
* 1. Import PR Demo project into Papyrus

The PR Demo project contains three projects,

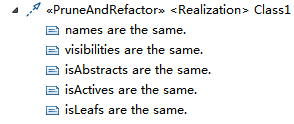


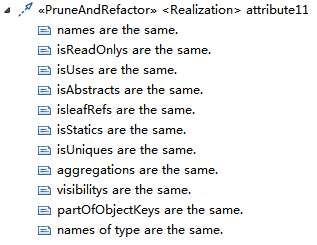


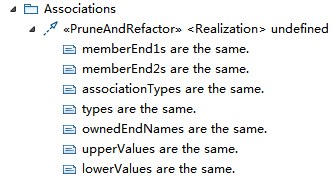
* 1. Drag pruning and refactoring realizations to the Diagram



* 1. Results in the comments of the PR Realizations

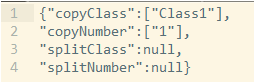




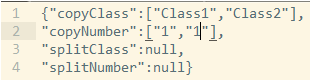


* 1. Copy “Class1” one time in the target model

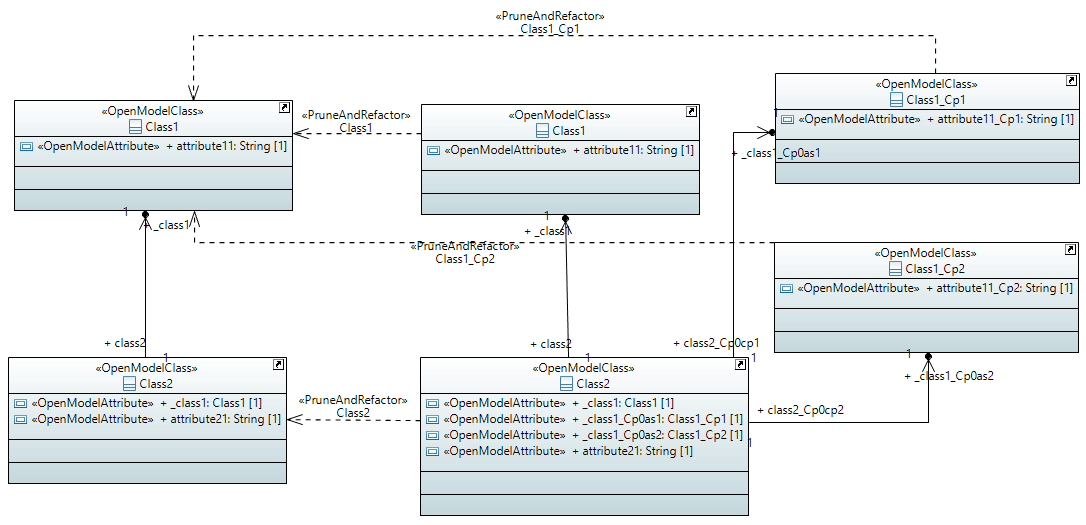
1. Edit the “CopyAndSplit.txt” as follows.



Note that you can copy multiple classes for any number of copies. For example, if you want to copy two classes, “Class1” and “Class2”, the content should looks like this.

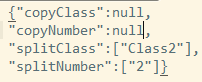


1. Move “CopyAndSplit.txt” into “project” folder
2. Run PR tool in the command line again.
3. Drag the PR realizations onto the Diagram and view the model.

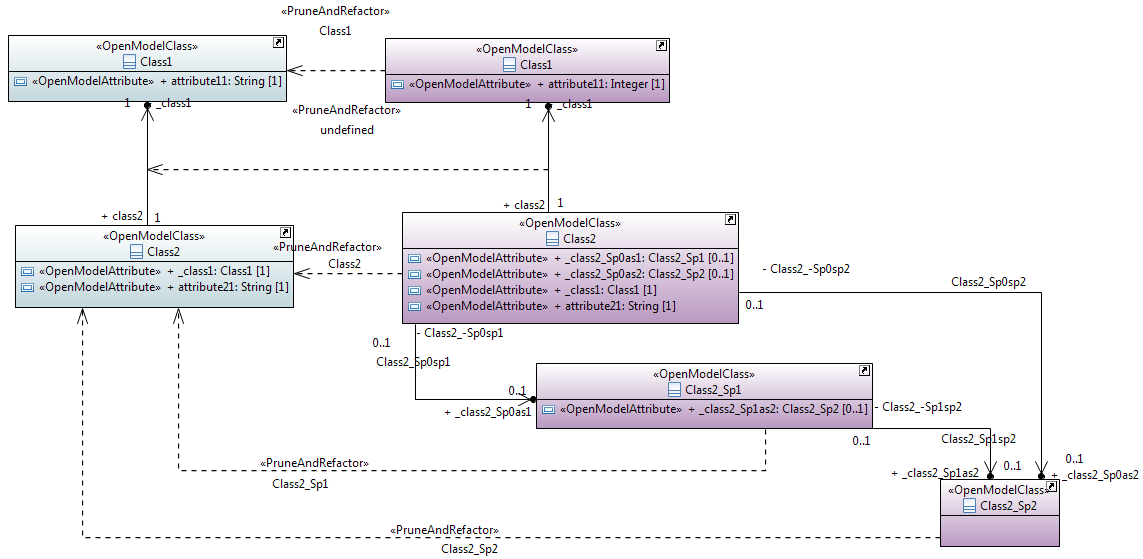


* 1. Split “Class1” one time in the target model

1. Edit the “CopyAndSplit.txt” as follows. Note that the user should set “null” to “copyClass” to prevent “Class1” to be copied again in the process of class split.



1. Do Step 2), 3) and 4) in Section 7.4. Then, “Class2” in the target will have a split class “Class2\_Sp1”.



# Naming Rules

1. For classes in the target model:

* Cloned classes will have the same name as in the source model;
* Copied classes will have the postfix “\_Cpi” in the names. For example, if the class needs be copied twice, the postfix of first copied class will be “\_Cp1”, and another will be “\_Cp2”;
* Splited classes will have the postfix “\_Spi” in the names. For example, if the class needs be splited twice, the postfix of first copied class will be “\_Sp1”, and another will be “\_Sp2”.

1. For attributes in the target model:

* Cloned attributes will have the same name as in the source model;
* Copied attributes will have the postfix “\_Cpi” in the names. For example, if the parent class of the subject attribute is copied twice, the postfix of the attribute is “\_Cp2”;
* Attributes from copied association will have the postfix “\_Cpiascpj”, in which “i” represents the “i”th copy of the association’s supplier and “j” represents the “j”th copy of the association’s client.
* Splited attributes will have the postfix “\_Spi” in the names. For example, if the parent class of the subject attribute is splited twice, the postfix of attribute will be “\_Sp2”.

1. For associations in the target model:

* Cloned associations will have the same name as in the source model;
* Copied associations will have the postfix “\_Cpicpj” in the names, in which “i” represents the “i”th copy of the supplier and “j” represents the “j”th copy of the client.
* Associations resulted from class split will have the postfix “\_SpiSpj” in the names, in which “i” represents the “i”th split of the supplier and “j” represents the “j”th split of the client.

1. For PR realizations in the mapping model:

* Cloned associations will have the same name as in the source model;
* Copied associations will have the postfix “\_Cpicpj” in the names, in which “i” represents the “i”th copy of the supplier and “j” represents the “j”th copy of the client.
* Associations resulted from class split will have the postfix “\_SpiSpj” in the names, in which “i” represents the “i”th split of the supplier and “j” represents the “j”th split of the client.

# Appendix I

Below is a table showing the xmi:id naming rules in the mapping model. The content of the table is organized according to the uml file structure. Each package is viewed as a folder in Papyrus.

