Appendix A

User Manual

This appendix introduces the installation of the AGSN editor and demonstrates the main features provided by the AGSN editor by showing the user's actions.

A.1 Installation

The AGSN editor is developed on Eclipse Modeling Tools Mars 1 Package. It can be installed on Eclipse Modeling Tools Mars 1 or 2. In the Eclipse, users can install new software by clicking on the menu "Help → Install New Software". To run our editor, users should install the following frameworks:

Epsilon Epsilon Update Site - http://download.eclipse.org/epsilon/updates/. Only the first four categories are needed: Epsilon Core, Epsilon Core Development Tools, Epsilon EMF Integration, Epsilon GMF Integration.

D-Case D-Case Editor Update Sites - http://dimensions-japan.org/dcase/eclipse/. If you do not see any package to install, you need to deselect "Group items by category" on the install wizard.

GMF This is installed via "Help → Eclipse Marketplace". Please search *Graphical Modeling Framework (GMF) - Tooling* and install it.

A.2 Features

In this appendix, the following four features are demonstrated: **Evidence management**, **Statistical report**, **ER score**, and **Model transformation**. For the features **Assessment status** and **Quality level**, please see the detailed introduction in Section 6.

A.2.1 Evidence management

Assume an example safety case is created in the AGSN editor (Fig. A.1). When a user select and right-click the evidence element *E_2*, a context menu will be shown. In the context menu "Attachment", three options are provided: "Select from Workspace", "Select from Web", and "Open".

By choosing the option "Select from Workspace", the editor will open a dialog as shown in Figure A.2. This dialog shows all the files under the example project. If an evidence attachment is saved as a file under the project, the user can select it and save it by clicking the button "OK". After that, the user can see that the property "Attachment" of the selected evidence element is changed to the path of the attachment.

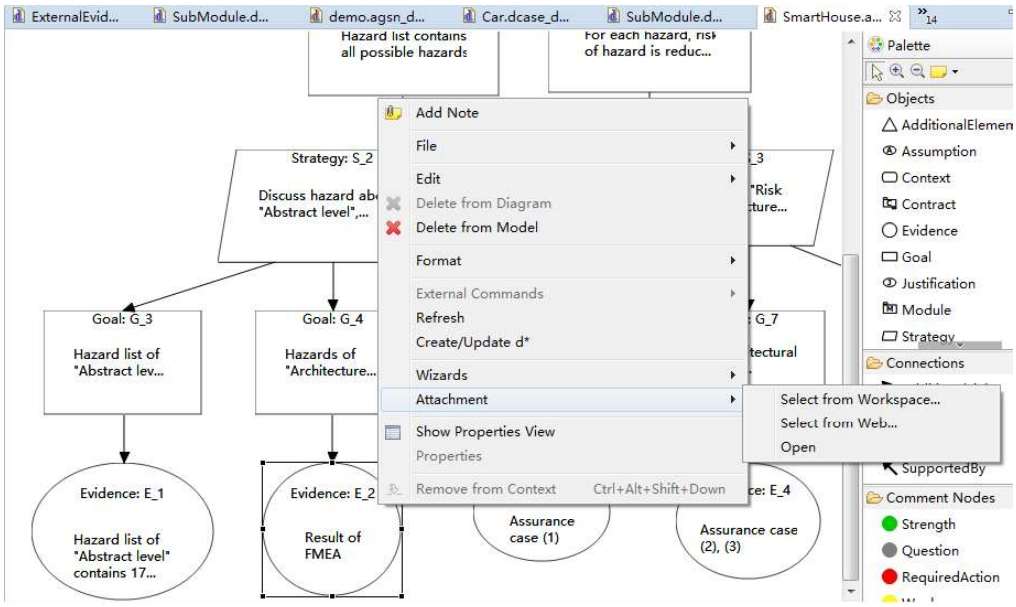

Figure A.1: Screenshot: Feature of Evidence Attachment in the AGSN Editor

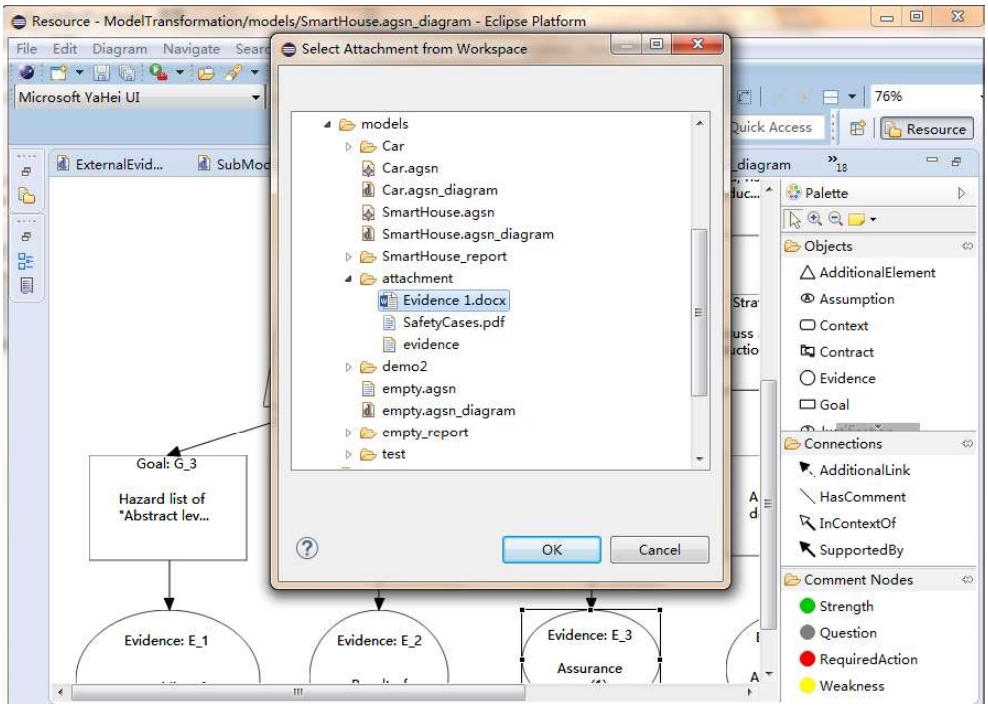

Figure A.2: Screenshot: Select Attachment From Workspace

By choosing the second option "Select from Web", the editor will open a dialog as shown in Figure A.3. The user can enter a URL in this text field. By clicking the botton "Browser", the editor will open the URL in web browser. By clicking the button "OK", the URL will be saved as the attachment of selected evidence element.

By choosing the third option "Open", the attachment will be opened depending on its type. If the attachment is a file in the workspace, it will be opened with its default editor, for example Adobe Acrobat Reader for PDF file. If the attachment is a URL, it will be opened in a web browser.

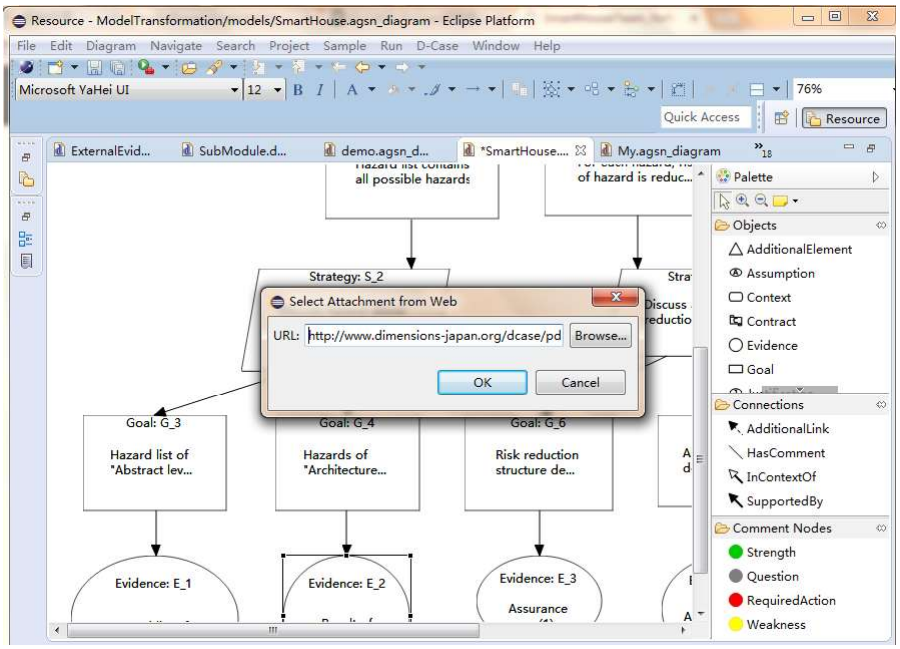

Figure A.3: Screenshot of AGSN editor: Select Attachment From Web

A.2.2 Statistical report

Assume an example safety case is created in the AGSN editor and it has been evaluated with the feature **Assessment status** and **Quality level**. The user can right-click on the canvas of a safety case, then a context menu will shown (Fig. A.4). By clicking the option "Quality Distribution", a quality distribution report will be generated and opened. By clicking the option "Assessment Status Overview", an assessment status overview report will be generated and opened.

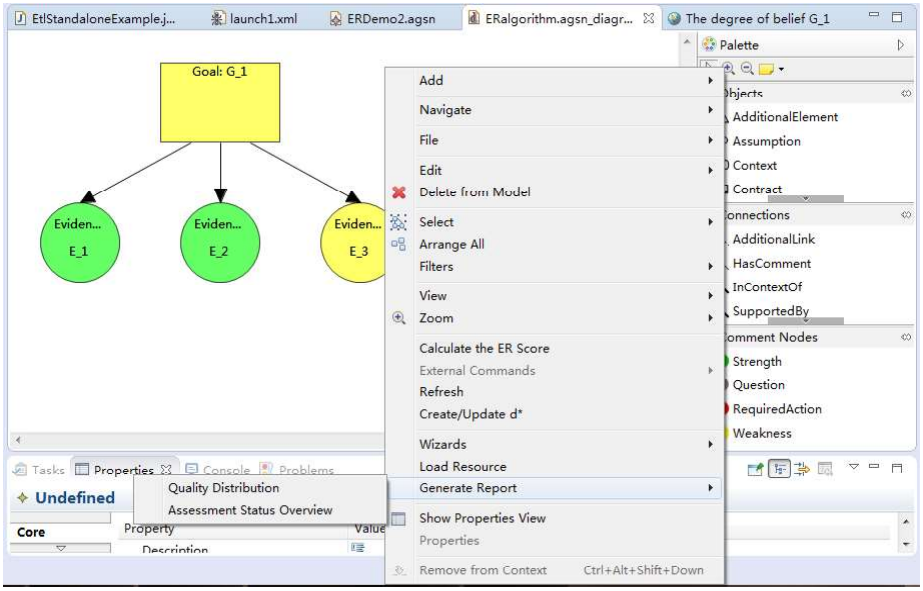

Figure A.4: Screenshot of AGSN editor: Generate Report Menu

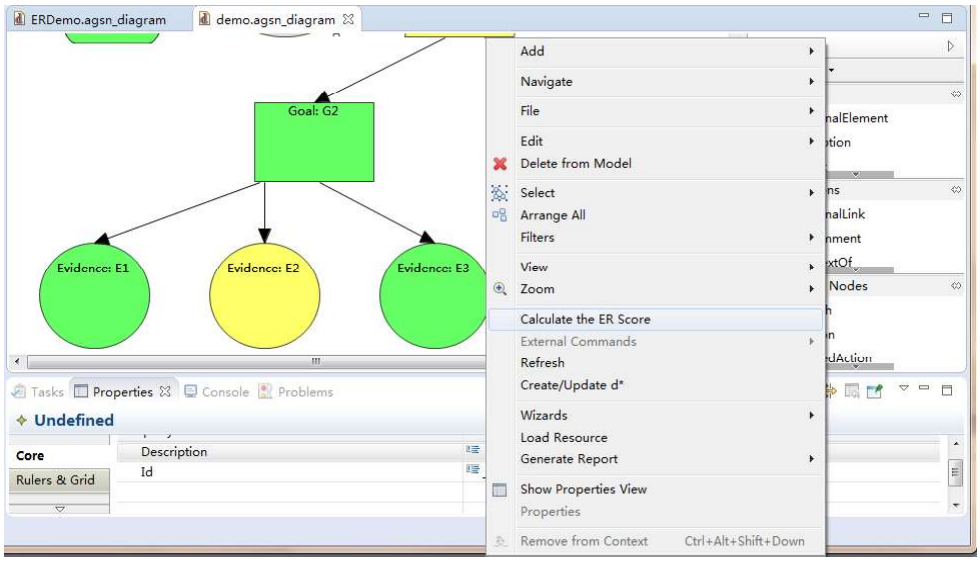

Figure A.5: Screenshot: Context menu of Calculating ER score

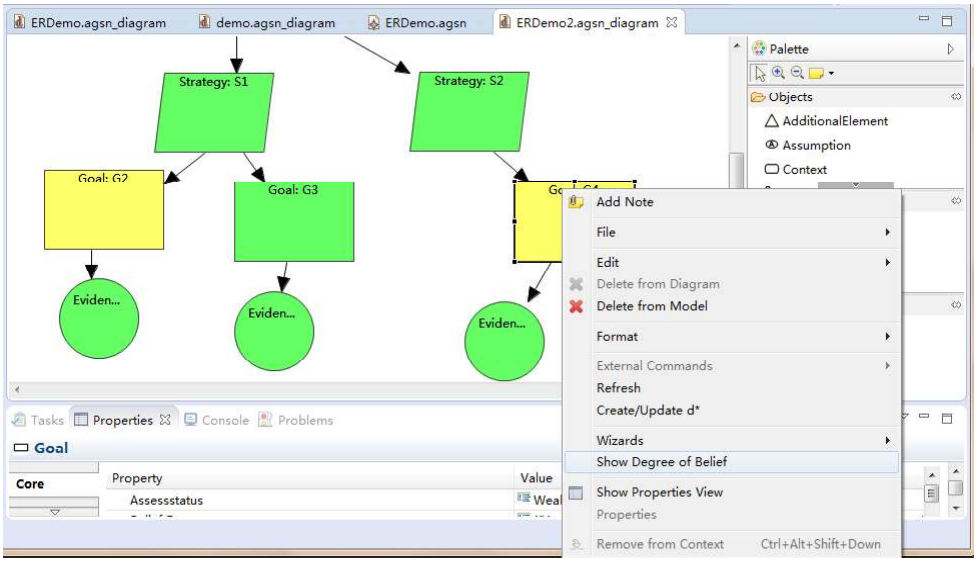

Figure A.6: Screenshot: Context Menu of Showing Degree of Belief

A.2.3 ER score

Assume an example safety case is developed and evaluated in the AGSN editor (Fig. A.5). The user can right-click on the canvas of this safety case, then a context menu will be shown. By clicking the option "Calculate the ER Score", the ER score of the safety case will be calculated.

After calculating the ER score, the ER score of each GSN element will be shown as the property "Belief Degree" in the properties view. To show the ER score in bar chart, the user can select an element and right click it, a context menu will be shown (Fig. A.6). By clicking the option "Show Degree of Belief", the ER score of selected element will be shown in a bar chart.

A.2.4 Model transformation

To transform a D-Case model to an AGSN model, the following preparation and configuration work should be performed. In this configuration, an Ant-build file is used to call the EGL, then the EGL can generate another Ant-build file to call the model transformation using ETL. Users should manually configure two files: the first Ant-build file and the EGL file before start transforming. List A.1 shows the configuration of the first Ant-build file.

Listing A.1: The location and name of the input model is configured in line 3-4. The values of these two properties should be configured by users. In line 5, the generated Ant-build file is located under the folder "scripts". The input D-Case model "Car.dcase_model" and output model "Car.agsn" is loaded to the EGL in line 6-9. In line 10-14, the EGL file is run on these two models and will output the file "launch.xml".

```
1  <?xml version="1.0"?>
2  <project default="main">
3  <property name = "sourcepath" value = "E:/TUEmaster/runtime/d-casedemo/"></property>
4  <property name = "sourcefile" value = "Car.dcase_model"></property>
5  <property name = "xmlfile" value = "../scripts/launch.xml"></property>
6  <target name="loadModels">
7  <epsilon.emf.loadModel name="dcase" modelFile="${sourcepath}${sourcefile}" metamodelUri="http://
       www.dependable-os.net/2013/11/dcase_model/" read="true" store="false"/>
8  <epsilon.emf.loadModel name="agsn" modelFile="../models/Car.agsn" metamodelUri="AGSN" read="true"
       store="false"/>
9  </target>
10 <target name="main" depends="loadModels">
11 <epsilon.egl src="../resource/etl/generateAntFile.egl" target = "${xmlfile}">
12 <model ref="dcase"/>
13 <model ref="agsn" />
14 </epsilon.egl>
15 </target>
16 </project>
```

Figure A.7 shows the EGL file used to generate the second Ant-build file. Users should configure the variable *sourcepath* and *modelname* as same as in the previous Ant-build file.

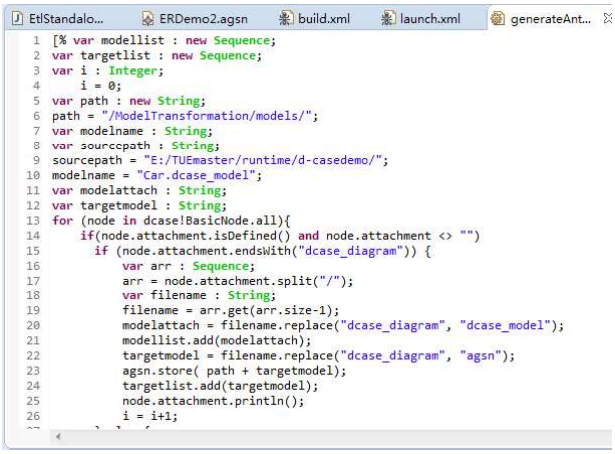

Figure A.7: Screenshot: Generate Ant-build file using EGL

After aforementioned two files are configured, users can run the first Ant-build file "build.xml" which is under the folder "scripts". Then the second Ant-build file is generated under the same folder. Listing A.2 shows the generated Ant-build file for the input model "Car.dcase_model". This file does not need to be configured and it can be use directly. After running this Ant-build file, the transformed AGSN models can be found under the folder "models".

Listing A.2: Line 6-11 is similar to the line 3-9 in the Listing A.1. Then in line 12-17, the input model "Car.dcase_model" is transformed to the model "Car.agsn". The attachment models "SubModule.dcase_model" and "SubModule.agsn" are loaded in line 19-23 and transformed in line 24-29.

```
1  <?xml version="1.0"?>
2  <project default="all">
3  <target name="all" depends="main
4  , attachment1
5  " />
6  <property name = "sourcepath" value = "E:/TUEmaster/runtime/d-casedemo/"></property>
7  <property name = "sourcefile" value = "Car.dcase_model"></property>
8  <target name="loadModels">
9  <epsilon.emf.loadModel name="dcase" modelFile="${sourcepath}${sourcefile}" metamodelUri="http://
       www.dependable-os.net/2013/11/dcase_model/" read="true" store="false"/>
10 <epsilon.emf.loadModel name="AGSN" modelFile="../models/Car.agsn" metamodelUri="AGSN" read="
       false" store="true"/>
11 </target>
12 <target name="main" depends="loadModels">
13 <epsilon.etl src="../resource/etl/DCaseToAGSNCopy.etl">
14 <model ref="dcase"/>
15 <model ref="AGSN" />
16 </epsilon.etl>
17 </target>
18 <target name="loadModels1">
19 <epsilon.disposeModel model="dcase"/>
20 <epsilon.disposeModel model="AGSN"/>
21 <epsilon.emf.loadModel name="dcase" modelFile="${sourcepath}Car/SubModule.dcase_model"
       metamodelUri="http://www.dependable-os.net/2013/11/dcase_model/" read="true" store="false"/>
22 <epsilon.emf.loadModel name="AGSN" modelFile="../models/Car/SubModule.agsn" metamodelUri="
       AGSN" read="false" store="true"/>
23 </target>
24 <target name="attachment1" depends="loadModels1">
25 <epsilon.etl src="../resource/etl/DCaseToAGSNCopy.etl">
26 <model ref="dcase"/>
27 <model ref="AGSN" />
28 </epsilon.etl>
29 </target>
30 </project>
```

Appendix B

Developer Manual

This appendix introduce the structure of the source code. It provides an overview of this project to help developers to understand the source code when they need to modify or extend this project. In this manual, we mainly discuss two parts of the project: the graphical editor and the dynamic model transformation.

B.1 The AGSN editor

The AGSN editor is developed using EuGENia and based on GMF. The structure of this project is a typical GMF project which contains projects like **.diagram*, **.edit*, **.editor* and **.test* (Fig. B.1). These projects are generated automatically. Besides, the project **.diagram.common* and **.diagram.custom* are developed by ourselves which contain the customized features and code of the AGSN editor. In the *GSN.figures*, the figures used to represent different GSN elements are drawn. If developers want to add or change the figures of the GSN elements, please add new figure class to this project. In the project *ModelTransformation*, developers can find the files used to transform the D-Case model to the AGSN model.

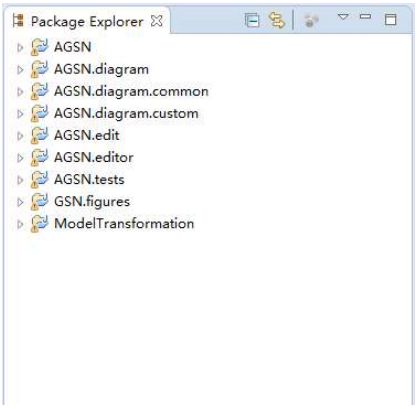

Figure B.1: Package Explorer View of the Project

Figure B.2 shows the structure of the *AGSN* project. The most important part is the ECore model and GMF models. To modify the AGSN metamodel, developers can either modify the ECore model or the ECore diagram. After that the *EMF* model can be generated from the ECore model. In the *EMF* model, the annotations are manually added to each elements in the ECore. These annotations are used by EuGENia to generate GMF models like *AGSN.genmodel*, *AGSN.gmfgen*, *AGSN.gengraph*, *AGSN.genmap* and *AGSN.gentool* automatically. Simultaneously, the source

code of the editor can be generated as well. For developers, they only need to modify the ECore model and the *EMF* model and other models and code can be generated.

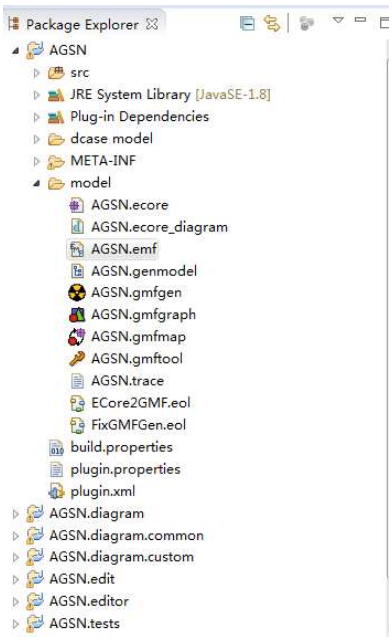

Figure B.2: Package Explorer View of the Project AGSN

Figure B.3 shows the structure of the *AGSN.diagram.common* project. This project is a customized plug-in project. In this project, a number of common objects are programmed to facilitate the customized features in the *AGSN.diagram.custom* project. The most Java classes in this plug-in is developed by D-Case. Usually, developers do not need to change these code. But it is possible to extend some attributes or values in these classes.

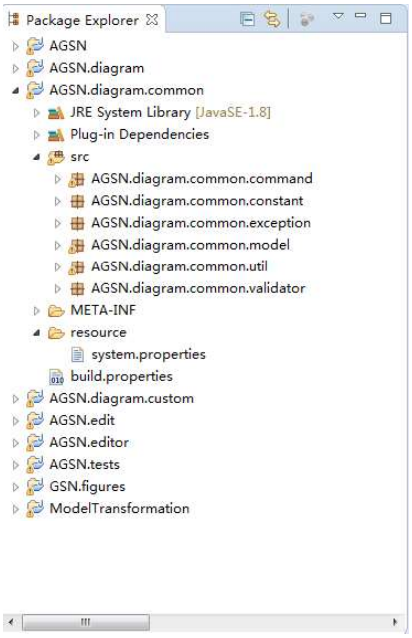

Figure B.3: Package Explorer View of the Project AGSN.diagram.common

There are six packages under the *src*: *command*, *constant*, *exception*, *model*, *util*, *validator*. The usage of these packages are as follows:

Table B.1: The packages in plug-in *common*

command	This package only contains one Java class which used to change the attribute of a node.
constant	All the constant values can be defined in this package for reuse.
exception	This packages contains the Java classes that construct the D-Case exception.
model	It represents the model of the GSN nodes and links. This package is not directly used by us.
util	This package contains a number of utilities provided by the D-Case, like file management, message management and link management, etc. The text of the message of listed in *messages.properties*.
validator	It contains the validation rule of D-Case model, this part is not directly used by us.

Figure B.4 shows the structure of the *AGSN.diagram.custom* project. This project is also a customized plug-in project. It contains the customized features developed for the AGSN editor. If developers want to extend the tool, the extension should be developed in this plug-in project. To develop new user action, developer need to add extension point in the *plugin.xml*. Many examples can be found in the GMF documentation or on the internet. Also developers can learn how to develop a new feature by looking to our existing features. In the *plugin.properties*, all the labels used by the extension point are listed.

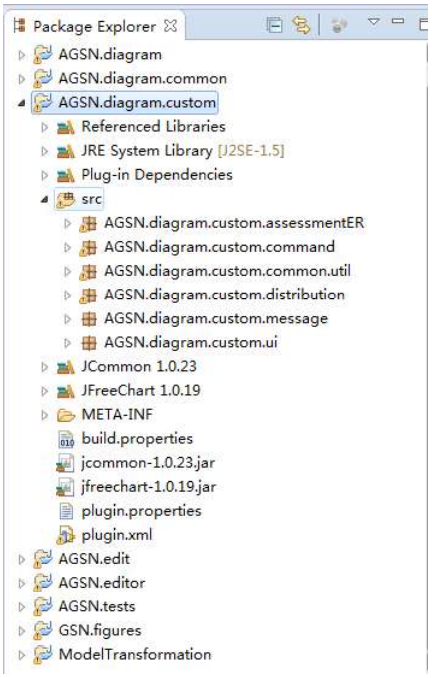

Figure B.4: Package Explorer View of the Project AGSN.diagram.custom

There are six packages under the *src*: *assessmentER*, *command*, *common.util*, *distribution*, *message*, *ui*. The usage of these packages are as follows:

Table B.2: The packages in plug-in *custom*

Package *assessmentER*	
Assessment	This is the class used to calculate the ER score of a safety case.
ChangeScoreTransctionCommand	This is the command used to change the ER score of element.
ERAlgorithm	It contains the calculation steps of the ER algorithm.
ERJFreeBarChart	This is used to generate ER bar charts.
ERTest	This is used to test the correctness of the ER algorithm.
Package *command*	This package contains all the handlers referred by the extensions in the plugin.xml. By adding new features in the editor, you may need to create new command, menu definition and handler class.
Package *common.util*	Three common utilities introduced by D-Case. It is not necessary to change these classes for development.
Package *distribution*	
ComputeNodesMember	This is developed by D-Case to count the number of nodes. It is used here to calculate nodes for statistical report.
ComputeNodesModel	This is used together with the previous Java class.
JFreeBarChart	This contains the generation of two statistical reports and it is called by quality and status distribution handlers in the package *command*.
Package *message*	This is developed by D-Case to show messages to users of the tool. It is not directly used by our tool.
Package *ui*	Three classes provided by D-Case to open web page dialog and show web pages in eclipse.

B.2 Model transformation

There are four main files used in this dynamic model transformation: *DCaseToAGSNCopy.etl*, *generateAntFile.egl*, *build.xml* and *launch.xml*. Initially, users should run *build.xml* to call the *.egl* which will generate the second Ant buildfile *launch.xml*. Then run the launch.xml will call the ETL *DCaseToAGSNCopy.etl* to transform the model and its attachment.

In the *build.xml*, developers can change the file path of the input model and the output Ant buildfile. Then the input and output models will be loaded. These models are used by the EGL file to generate the corresponding Ant buildfile.

To modify the content in the generated Ant buildfile, developers should change the output text in blue. Other EOL codes (inside [%...%]) are used to read the model and detect all the attachment. The EOL will find all the attachments referred by the **BasicNode** and the **BasicLink**. Because the name of the attachment is not always ended with correct file extension. Therefore, we need parse the name and replace the extension by *.dcase_model*. To modify the model transformation, developers should change the rules in the *DCaseToAGSNCopy.etl*. The Ant buildfile *launch.xml* is generated using EGL, so this file does not need to be modified.

Under the folder *src*, a number of examples are used to run the ETL using Java code, but it is

not succeed. For interested developers, you can try to solve the problem.

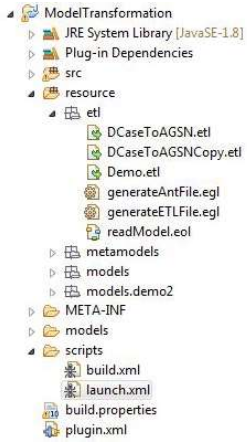

Figure B.5: Package Explorer View of the Model Transformation