1. MatchAnyAssoc Example

This example demonstrates the match-any-association feature in GReAT. The block TestAssociationClass demonstrates how the feature works for matching association relationships that involve an association class. The block TestSimpleAssociation demonstrates how the feature works for matching simple association relationships (that do not involve an association class). Every rule counts the number of matches found by setting various counter attributes of ClassA objects. Every rule contains annotations describing the resulting number of matches, and you can verify the results by examining the generated counter values in the output model, UmlModel_model.xml.

Please refer to the GReAT user manual for more information.

1.1. Directory Organization

- **MatchAnyAssoc**
  - testMatchAnyAssoc.mga - The transformation file
  - testMatchAnyAssoc.xme - transformation exported
  - **Meta**
    - BidirectAssoc.mga - metamodel that consists a bidirectional association
    - BidirectAssoc.xme - the above metamodel exported to XML
    - GS_BidirectAssoc.xmp - BidirectAssoc paradigm file
  - **Models**
    - BidirectAssoc_model.mga - model of BidirectAssoc paradigm
    - BidirectAssoc_model.xme - Example model exported
  - **Udm** - Will contain the Udm meta files
  - **Gen** - Will contain the CG tool generated files

1.2. How to run MatchAnyAsso example?

Open MatchAnyAssoc transformation model

- Directly open $/testMatchAnyAssoc.mga, if it fails, open GME, choose File/Import XML, and choose $/testMatchAnyAssoc.xme

testMatchAnyAssoc.mga contains the transformation rules, UDM compatible meta information paradigms and configuration information. Following is the folder structure which is shown in browser:

▪ testMatchAnyAssoc		
o GS_BidirectAssoc	-	Input Metamodel in UML class diagram format
o UMLModel	-	Output Metamodel in UML class diagram format
o NewTransformation	-	Folder containing the transformations
o NewConfigurations	-	Folder containing configuration information

Run the MatchAnyAssoc transformation model

- Invoke the GReAT Master Interpreter with icon  (**This is a required step for the first time running**). Use the default file paths and names provided.
- The transformations can be invoked in various ways
    1. GR Engine – Performs the transformations in an interpretive manner
    2. GR Debugger – Provides a user interface and debugging features such as break points, single step, step into etc.
    3. Code generator – Converts the transformation into code that can be compiled and executed.
- To run GR Engine, it could be done either :
    - In the GReAT Master Interpreter dialog, check the box "Run GR Engine"-**or-**
    - Directly invoke the GR Engine interpreter with icon  .
    - The default input file is $/Models/BidirectAssoc_model.mga
    - The output files will be $/Models/UmlModel_model.xml
- To run the GR Debugger
    - Open a command prompt and go to the sample directory $/. Invoke GRD by calling GRD.exe , then load the config file $/config.mga **-or-**
    - Directly invoke the GR Debugger interpreter with icon  .
- To run Code Generator, it could be done either :
    - In the same dialog of GReAT Master Interpreter, check the box of "Run Code Generator"; **-or-**
    - Directly invoke the Code Generator interpreter with icon  ; **-or-**
    - Open a command prompt and go to the sample directory $/. Invoke CG by calling CG.exe , with config file $/config.mga
    - After the files have been generated open the generated Visual Studio project using VS71 and compile the project
    - You can run the generated code with default arguments by setting the working directory to be ..\ and Program argument to be –d (default)