1. SignalFlowGroup Example

This example demonstrates the group-match feature in GReAT. The rule *GroupingRule* demonstrates how the feature group-match works. In this rule, there is a Group element which contains PrimitiveComponent *PC1* and *PC2*, and *Signa*l association as its members. After all matches found, the matches will be grouping into different subgroups according to the criteria defined in the Group attribute "*Criteria for grouping matches*". For each group of matches, there is a new CompoundComponent *newCC* will be created, and then move all objects in the group into this new created parent object. Finally applying the AttributeMapping code will set a new name to the CompoundComponent *newCC.*

Please be careful about the format of the code in the AttributeMapping in the rule which contains Group element. Since a rule with Group is not restrained to the single bindings as other rules, the PatternClass name doesn't represent single object in the code, but the list of objects bound to the PatternClass.

Please notice that currently group-match feature is only supported by GRE.

Please refer to the GReAT user manual for more information.

1.1. Directory Organization

- **SignalFlowGroup**
    - MoveSignalFlowGroup.mga - The transformation file
    - MoveSignalFlowGroup.xme - transformation exported
    - **Meta**
        - **Icons** - Icons for the GS_SignalFlowGroup paradigm
        - GS_SignalFlowGroup.mga - GS_SignalFlowGroup metamodel
        - GS_SignalFlowGroup.xme - the above metamodel exported to XML
        - GS_SignalFlowGroup.xmp - paradigm file
    - **Models**
        - mySignalFlow1.mga - sample model
        - mySignalFlow1.mga - above sample model exported
    - **Udm** - Will contain the Udm meta files

1.2. How to run SignalFlowGroup example?

Open SignalFlowGroup transformation model

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

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

▪ MoveSignalFlowGroup		
o GS_SignalFlowGroup	-	UML Metamodel class diagram format
o zt_MoveSF	-	Folder containing the transformations
o zz_Config	-	Folder containing configuration info

Run the MoveSignalFlowGroup 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/mySignalFlow1.mga
  - The output files will be $/Models/my_newSignalFlow.mga
- 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 .