

Expressing Goal-oriented Requirement
Language in UML 2.0:
Examining the functionality of UML Profiles

April 2005

Nadir Janmohamed
Supervising Professor: Daniel Amyot
SITE, University of Ottawa

Table of Contents

1	<i>Introduction.....</i>	3
2	<i>Scope.....</i>	4
3	<i>Profiles in UML 2.0</i>	5
3.1	Stereotype Mechanism	5
3.2	Metamodel Extension Mechanism	5
4	<i>Implementing GRL notation in UML Modeling Software Tools.....</i>	6
4.1	IBM Rational Architect	6
4.1.1	GRL Profile in RA	6
4.1.2	Usability of RA as a GRL modeling tool	8
4.1.3	Limitations of RA.....	9
4.2	Telelogic TAU G2	9
4.2.1	GRL Profile in TAU I: Stereotypes extension.....	9
4.2.2	GRL Profile in TAU II: Metamodel extension.....	10
4.2.3	Usability of TAU as a GRL modeling tool.....	11
4.2.4	Limitations of TAU	11
5	<i>Discussion: The feasibility of implementing all of GRL in UML Profiles</i>	12
6	<i>Discussion: The feasibility of implementing UCM in UML Profiles.....</i>	12
7	<i>Conclusion.....</i>	13
	<i>Appendix A: Rational Software Architect.....</i>	14
A1:	How to create a profile in RA.....	14
A2:	How to use a Profile in RA	15
	<i>Appendix B: TAU G2.....</i>	16
B1:	How to create a simple profile in TAU	16
B2:	How to activate and use a simple profile in TAU	17
B3:	How to create the structure for a (metamodel) profile in TAU.....	17
B4:	How to use a (metamodel) profile in TAU.....	21
B5:	Reference (metamodel) Profiles in Tau	21
	<i>Reference:.....</i>	22

1 Introduction

The purpose of this document is to explore the use of UML 2.0 Profiles to express GRL (Goal-oriented Requirements Language). UML is a powerful modeling language that can be used to model a wide range of systems. However, UML does not provide an especially powerful way to represent and analyze requirements. Representing and analyzing requirements in a model can be very useful in the early stages of any design process. GRL is a modeling language designed to express and analyze requirements with a special emphasis on non-functional aspects. Obviously, GRL cannot be used alone to model an entire system but it is intended for GRL to be used in conjunction with other modeling languages/notations, in fact GRL is a component of a notation called User Requirement Notation (URN). UML 2.0 provide an extension mechanism through UML Profiles that be used to customize the UML metamodel to represent specific domains and or other modeling languages/notations.

The study presented in this document examines the feasibility of using UML Profiles to express GRL. In this study, two different modeling software applications are used (both are UML 2.0 compliant):

- IBM Rational Architect 6.0 (RA)
- Telelogic TAU G2 2.4

Both of these applications provide support for UML Profiles. Profiles representing GRL were built in both of these applications. This document describes how these profiles were built and it also compares the quality of these profiles with respect to the actual GRL notation.

2 Scope

This document discusses how UML Profiles were used to represent GRL in IBM Rational Architect and Telelogic TAU G2 2.4. For the purpose of this document, GRL can be thought of as having two major components:

1. GRL notation – the set of symbols and lines used to actually model requirements
2. GRL Evaluations – the mechanism (algorithm) in GRL used to examine the satisfaction levels of the requirements

This document focuses on the first component: the GRL notation. The GRL notation is what was implemented in the UML Profiles. This study does not cover GRL Evaluations. However this document does briefly discuss how GRL Evaluations could be implemented to work with UML Profiles.

As mentioned in the introduction, GRL is a component or User Requirement Notation (URN). This document does not discuss the other aspects of URN in any great detail. However this document does briefly discuss URN and its possible relationship with UML Profiles.

It is assumed that the reader of this document is comfortable with GRL¹, URN², UML, and the software application mentioned above.

¹ For a full description of GRL refer to the GRL website

² For a full description of URN refer to the URN website

3 Profiles in UML 2.0

Profiles in UML 2.0 are an extension mechanism that can be used to customize the UML metamodel for various purposes. It is important to note that Profiles can be used to adapt/extend the UML metamodel, but this mechanism does not allow the UML metamodel to be directly modified³. As mentioned earlier, for the purpose of this study UML Profiles were used to represent the GRL notation.

In this study there were two different ways Profiles were used to represent GRL:

1. GRL Profile created using the Stereotype Mechanism
2. GRL Profile created using the Metamodel Extension Mechanism

Both of these mechanisms are described below.

3.1 Stereotype Mechanism

The stereotype mechanism is a very straightforward way of customizing UML. Simply speaking, a *stereotype* is an extension of a basic UML element that contains simple customizations (i.e., custom name, custom attributes, custom appearance etc...). For example a GRL Task element could be represented as a stereotype of a UML Class. Similarly a GRL Correlation link could be represented as a stereotype of UML Association. Now the actual GRL Profile constructed in this manner would simply be a set of stereotypes. This set of stereotypes can be used to create GRL diagrams inside UML models.

This type of Profile is very easy to construct however it suffers from some limitations. Since all the GRL elements are just stereotypes of existing UML elements, they can only be used in regular UML diagrams. This means that other non-GRL elements could be mixed with the GRL diagrams, which could be very confusing. Also, this type of Profile is not necessarily very user-friendly for the designer who wishes to use this GRL profile to create a GRL model. All of these limitations are discussed in detail in section [4 Implementing GRL notation in UML Modeling Software Tools](#).

3.2 Metamodel Extension Mechanism

The metamodel extension mechanism is a more robust extension mechanism that has all the functionality of the stereotype extension mechanism as well as the ability to customize non-basic UML elements such as diagrams. For example a GRL Diagram can be represented as a metaclass extension of UML Class Diagram and then restrictions on the GRL Diagram could be place to allow only GRL elements.

This type of Profile is more complex to implement. It requires the UML metamodel to be extended and then customized/restricted to meet the needs of the GRL notation. There are limitations with this extension mechanism also. These limitations are presented in section [4 Implementing GRL notation in UML Modeling Software Tools](#).

³ For the complete definition of a UML Profile refer to the UML 2.0 Infrastructure Specification

4 Implementing GRL notation in UML Modeling Software Tools

This section of the document describes how GRL Profiles were created and deployed in IBM Rational Architect and Telelogic TAU G2. This section also discusses and compares the usability of the created GRL Profiles in each application

4.1 IBM Rational Architect

IBM Rational Software Architect (RA) is an integrated software development environment that is UML 2.0 compliant. RA provides a stereotype mechanism for UML Profiles. Appendix A of this document describes how to create and use a simple profile. The specific tasks described in Appendix A were used to create a GRL Profile in RA. The section below describes which GRL element stereotypes created and their associated metaclasses.

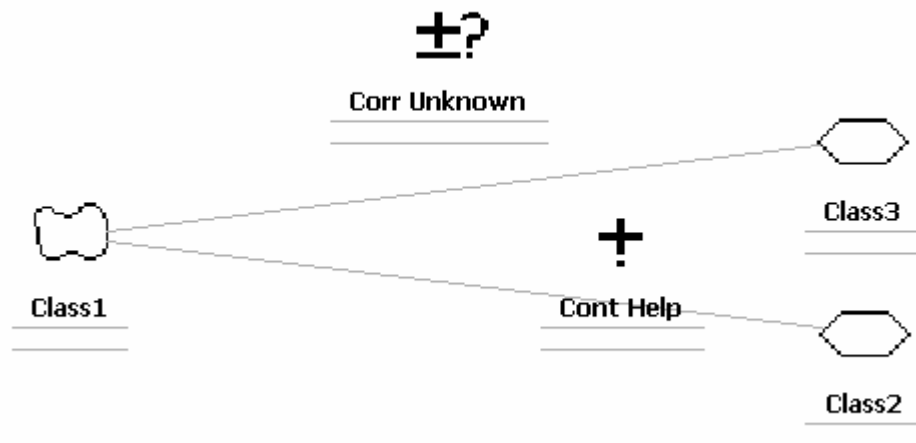
4.1.1 GRL Profile in RA

The GRL Profile created in RA is fairly straight forward. The intentional elements in GRL are stereotypes of the UML Class metaclass. GRL links are stereotypes of UML Association metaclass or the Association Class metaclass. The actual GRL diagram was simply a UML class diagram with the extra GRL stereotypes. The Actor Boundary in GRL was omitted in this profile because there did not seem to be a way to represent this element in a UML Class diagram.

For the intentional elements in GRL custom icons and shapes were used (a simple icon creation application was used to generate the appropriate shapes).



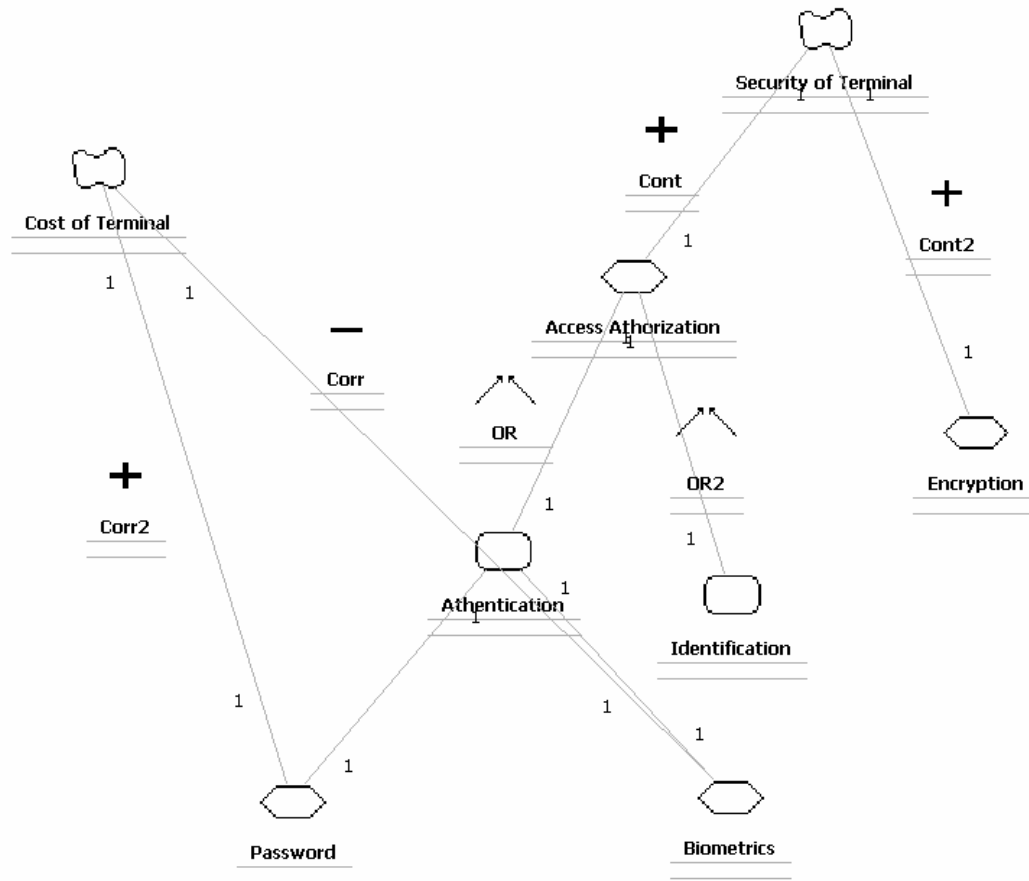
RA doesn't provide a mechanism for customizing link styles so GRL links all appear similar. For Contribution and Correlation links Association Class links were used (this way the icons describing the contribution/correlation types could be used)



The table below summarizes the GRL stereotypes and their associated metaclasses that were used in the RA GRL Profile.

GRL Intentional Elements	Metaclass (stereotype extension)	Custom Appearance	GRL Links	Metaclass (stereotype extension)	Custom Appearance
Goal	Class	Yes (Shape)	Contribution	Association Class	Yes* (type only)
SoftGoal	Class	Yes (Shape)	Correlation	Association Class	Yes* (type only)
Belief	Class	Yes (Shape)	Means End	Association	No
Resource	Class	Yes (Shape)	Dependency	Association	No
Task	Class	Yes (Shape)	Decomposition	Association	No
Actor	Class	Yes (Shape)			
Actor Bound	N/A	N/A			

Here is an example of a GRL diagram in RA.



4.1.2 Usability of RA as a GRL modeling tool

In terms of usability RA, is not very strong. Creating the profile was fairly straightforward but using the created profile in a model was not very user-friendly. If the GRL stereotype was an extension of the UML Class metaclass then the palette provided an easy way of using this stereotype by clicking on the Stereotyped Class icon and then selecting the desired stereotype from a list. However for stereotypes that did not extend the UML Class metaclass (such as GRL Correlation, which extended the Association Class metaclass) these stereotypes would have to be applied manually using the Properties view, this method was very tedious.

4.1.3 Limitations of RA

- RA only supports one type of UML Profile: Stereotype Mechanism
- The appearance of UML Links cannot be customized in RA
- RA does not support custom restrictions on the end points of UML links
- Actor Boundary's cannot be expressed in RA
- RA does not support the creation of custom diagram types
- RA does not support modeling UI customization directly through profiles (however RA does allow the UI to be customized via its eclipse based Java API, but this was beyond the scope of this study)

4.2 Telelogic TAU G2

Telelogic TAU G2 is UML 2.0 based tool that allows for software to be designed in Model Driven Approach. TAU supports both types of UML Profiles: stereotype extension and metamodel extension.

4.2.1 GRL Profile in TAU I: Stereotypes extension

Creating a Profile of this type is fairly straightforward in TAU. A Profile project is created and then stereotypes that represent GRL elements are added to the profile and the Profile is activated. For full details refer to Appendix B1 and B2.

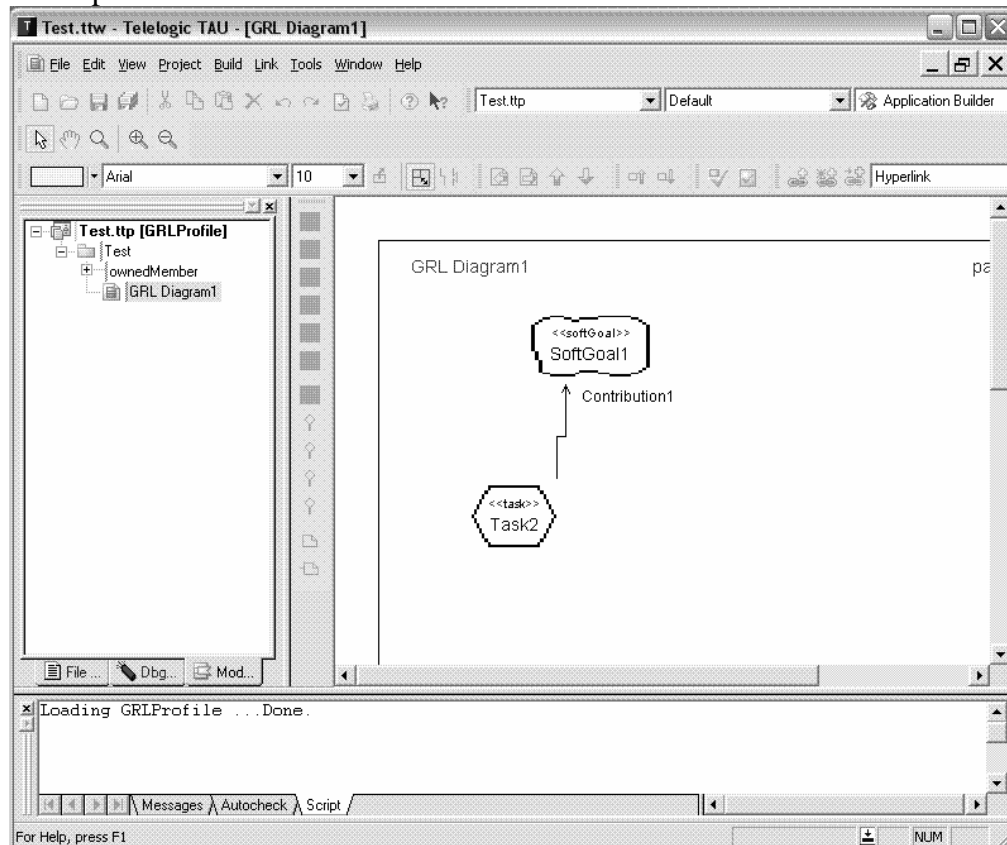
Specifically, intentional elements in GRL are stereotypes of the UML Class metaclass. GRL links are stereotypes of UML Association metaclass. Once again due to the limitation of this mechanism Actor Boundary's are not expressed in this Profile

The table below summarizes the GRL stereotypes and their associated metaclasses that were used in the TAU *simplegrlprofile*.

GRL Intentional Elements	Metaclass (stereotype extension)	Custom Appearance	GRL Links	Metaclass (stereotype extension)	Custom Appearance
Goal	Class	Yes (Shape)	Contribution	Association	No
SoftGoal	Class	Yes (Shape)	Correlation	Association	No
Belief	Class	Yes (Shape)	Means End	Association	No
Resource	Class	Yes (Shape)	Dependency	Association	No
Task	Class	Yes (Shape)	Decomposition	Association	No
Actor	Class	Yes (Shape)			
Actor Bound.	N/A	N/A			

4.2.2 GRL Profile in TAU II: Metamodel extension

Tau also provides a much more robust metamodel extension Profile. This Profile type in TAU allows for custom diagram types to be created, also it allows modeling UI to be customized as well as providing all the functionality that the stereotype extensions provides.. Using this mechanism a custom GRL Diagram type was created along with a customized palette and model view.



Due to the large number of metaclasses in this GRL Profile, only the outline of how this GRL Profile was created is described. This description is in Appendix B3 and B4. For full details refer to the diagrams in the actual GRLProfile itself.

Basically there were three main sets of components in GRL metamodel.

- The Metaclasses representing the UML components available in the GRL Profile (i.e., Class, Association and there associated metaclasses)
- The metaclasses representing the UML class diagram and the associated symbols and lines
- The metaclasses representing the actual GRL elements (i.e., the GRL intentional elements and the GRL links)

The diagrams in the actual GRLProfile display how the GRL profile was constructed. Also the properties of the metaclasses in the model list the necessary values that were needed to make the GRL Profile work (See Appendix B3).

In this Profile, the actual GRL elements are represented by metaclasses as well as stereotypes. Each GRL element is has a base metaclass as well as a base stereotype (the stereotype is of the same name). This aspect is very similar to *simplegrlprofile*.

GRL Intentional Elements	Base Metaclass	Custom Appearance	GRL Links	Base Metaclass	Custom Appearance
Goal	Class	Yes (Shape)	Contribution	Association	No
SoftGoal	Class	Yes (Shape)	Correlation	Association	No
Belief	Class	Yes (Shape)	Means End	Association	No
Resource	Class	Yes (Shape)	Dependency	Association	No
Task	Class	Yes (Shape)	Decomposition	Association	No
Actor	Class	Yes (Shape)			
Actor Bound	N/A	N/A			

4.2.3 Usability of TAU as a GRL modeling tool

Using the stereotype mechanism Profile

The *simplegrlprofile* is fairly straightforward to use. This profile is simply applied to a UML model and then the GRL diagram is created on a class diagram. The UI allows the GRL stereotype elements to be applied easily (for example, right click on a UML Class in the class in diagram and click Stereotypes then check Goal). The UI experience of this *simplegrlprofile* is slightly better than RA.

Using the metamodel mechanism Profile

The GRLProfile has a very User Friendly Interface. Once the Profile is applied a user can create a GRL Diagram and then add GRL elements directly from the customized palette (only GRL elements are shown on the palette).

It is important to note that creating this type of Profile requires a considerable more amount of work than a simple profile.

4.2.4 Limitations of TAU

- The appearance of UML Links cannot be customized in TAU
- TAU only supports custom restrictions on the end points of UML links using the TAU Agents which are not a part of the UML Profile definition (this will be discussed in section 5)
- Actor Boundary's cannot be expressed in TAU

5 Discussion: The feasibility of implementing all of GRL in UML Profiles

There are two main components of GRL: the GRL Notation and the GRL Evaluation mechanism. The following will highlight issues in implementing GRL in UML Profiles

Problems expressing the GRL Notation in UML:

It seems RA and TAU have some common problems in expressing the GRL notation:

1. The End point of GRL links cannot be restricted within the Profile Mechanism
2. The appearance of the GRL links cannot be customized (this is that big of a problem)
3. Actor Boundary's cannot be represented in Profiles

TAU does provide the TAU Agent mechanism (TAU Agents will not be fully supported until TAU G2 2.5 which allows external Com tools to interact with model elements. TAU Agents could possibly be used to solve the first problem. However both TAU and RA do not readily have ways to solve the second and third problems. In fact, the UML Infrastructure specification clearly states the Profiles are not a first class extension mechanism (meaning that existing UML metamodels cannot be directly changed but instead only adapted/extended). So, to solve the third problem, an Actor Boundary would have to be expressed as some element in a class diagram or at least as an UML element that could contain a class diagram. In any case, some type of compromise of the GRL notation would have to be made to make Actor Boundary's work in UML Profiles.

Problems expressing GRL Evaluations in UML:

Unfortunately there is no part of a UML Profile that has a construct that can be adapted to actively represent a GRL Evaluation. In order for GRL Evaluations to work with UML, some external mechanism would have to interact with the UML Profile. Once again TAU Agents could possibly be used but this would a propriety solution.

The Problems of fully expressing GRL in UML seem to indicate that only the GRL notation can be partially expressed and used with in UML Profiles at this point in time.

6 Discussion: The feasibility of implementing UCM in UML Profiles

Use Case Maps (UCM) are the other major part of URN. They are a considerably more complex than GRL. Unless UCM's could be adapted to fit congruently with a UML concepts (e.g., Activity Diagrams) it seems that implementing UCM's in UML Profiles would not be very fruitful, at least from a usability point of view.

7 Conclusion

This study has shown that it is possible to express the GRL notation in UML Profiles with some limitations. This study has also revealed that at the moment implementing all of GRL (Notation and Evaluations) is not feasible with the current UML tools available if the goal is to use these tools as GRL model editors.

Also after examining the actual UML Profile specification, it seems that the Profile mechanism is not powerful enough to express all of GRL.

Appendix A: Rational Software Architect

A1: How to create a profile in RA

This section describes to create and deploy a simple UML Profile Project in RA. For more information see the help documents (ie File → Help) in RA.

Create a UML Profile Project:

1. In RA, go to the menu File → New → Project ...
2. In the New Project dialog, expand Modeling → UML Extensibility and select UML Profile Project and click Next
3. Type in the name <My Profile> of the Project and click Finish

Add a Stereotype to the Profile:

1. In the Model Explorer expand the UML Profile Project <My Profile>
2. Right click on the Profile folder, Add UML → Stereotype
3. Type in the name of the stereotype <mystereotype>
4. Select the newly created stereotype <mystereotype> in the Model Explorer
5. In the Properties view (at the bottom of the window) click on the Extensions tab
6. Click on Add, then select in a metaclass to be extended in the dialog (for this example choose Class) and click ok
7. Save the changes made

Optional: Specify an icon and image for a stereotype

(Not available for UML links such as Associations, Dependencies ...)

1. Select the stereotype <mystereotype> in the Model Explorer
2. In the Properties view (at the bottom of the window) click on the General tab
3. Click on the Browse button next to Icon field
4. Navigate to the location of an icon image file (.ico this can be created using a icon creating/drawing application) and click open
5. Click on the Browse button next to Shape field
6. Navigate to the location of an shape image file (.ico) and click open
7. Save the changes made to the Profile.

Deploy the Profile:

1. After creating the Profile and the changes have been saved, right click on the Profile folder and select Release...
2. Type in a release label (such as 1.0 or a short description) and click ok
3. The Profile is now ready to use in a UML model.

A2: How to use a Profile in RA

This section describes how to use a Released Profile in a UML Model Project in RA.

Create a UML Model Project

1. In RA, go to the menu File → New → Project ...
2. In the New Project dialog, select UML Project and click Next
3. Type in the name <My Model> of the Project and click Finish

Add the Profile to the Model Project:

1. In the Model Explorer, expand <My Model> and select the element called Blank Model (one level below the Blank Model.epx)
2. In the Properties view click on the Profiles tab
3. Click on Add Profile... and in the dialog select File and Browse to the Profile file name (<My Profile>.epx) and then click ok

Optional: Specify Custom Shapes to be seen on diagram

1. Go to the menu Windows → Preferences
2. In the dialog, expand Model → Appearance and select Shapes
3. Click on the drop down for Stereotype Style and select Shape Image

Use an element in the Profile on a diagram:

1. Create a diagram in the Model (or open the default freeform diagram)
2. In the Palette view (on the right part of the window) expand the Class Diagram elements
3. Click on the drop down arrow next to the Class element and select Stereotyped Class
4. Place the Stereotyped Class on the diagram by clicking on the diagram canvas.
5. A list of stereotypes should appear, select the stereotype <mystereotype>

Note: For stereotypes of metaclasses other than the UML Class metaclass, the application of the stereotypes has to be done manually by first creating a normal version of the UML element in the diagram and then manually applying the stereotype by using the Properties view for that element (ex for a Correlation link a regular Association Class would have to be created and then the stereotype of a GRL Correlation would have to be applied using the Properties view for this element)

Appendix B: TAU G2

B1: How to create a simple profile in TAU

This section describes how to create a profile using the stereotype extension mechanism in TAU⁴

Create the Profile Directory Structure:

1. Using a Windows File Explorer, Navigate to the folder <Tau Installation>\addins
2. In this folder create a directory and call it simplegrlprofile
3. In simplegrlprofile directory create two directories etc and script
4. Also in simplegrlprofile create a text file called simplegrlprofile.mod that should look similar to the one below

```
[simplegrlprofile]
"scope"="PROJECT"
"version"="1.0"
"longname"="Simple GRL Profile"
"description"="Provides GRL Modeling functionality."
"product"="elvis"
[simplegrlprofile/Bin]
"listBin"=""
[simplegrlprofile/Script]
"listScript"=""
[simplegrlprofile/Etc]
"listEtc"="urn:u2:addins/simplegrlprofile/etc/simplegrlprofile.u2"
```

Create the Profile Project in TAU:

1. Launch TAU G2, and create a UML Modeling Project named simplegrlprofile in the etc directory created above
2. Expand the Model, and right click on the package and select Stereotypes...
3. In the Dialog check TTDPrefinedStereotypes:profile

Create a Stereotype:

1. Add a class diagram to the simplegrlprofile model
2. Expand the Library → TTDMetamodel and find the element called Class
3. Drag this element on to the diagram
4. Create a stereotype on the diagram and call it Goal
5. Create an extension link between the Goal Stereotype to the TTDMetamodel::Class

(Optional) associate an icon and an image for the Stereotype

6. Right click on the Goal Stereotype and select Stereotype...
7. In the dialog check TTDStereotypeDetails::icon and click ok
8. Right click on the Goal Stereotype and select Properties
9. In the Edit Properties view select Icon from the Filter drop down

⁴ This description is based on the white paper *How to Create a Profile in Tau G2* by Greg Gorman

10. Click on the ... button next to the Icon File field and navigate to the location of icon file for the stereotype (.ico)
11. Save the changes the model

B2: How to activate and use a simple profile in TAU

Create a UML Modeling Project and activate the profile

1. In TAU create a new UML Modeling Project call it Sample
2. Go to the menu Tools→Customize
3. In the Customize dialog, select the Add-ins tab
4. Check the box labeled simplegrlprofile and click close

Using the Profile

1. Create a new class diagram in the model
2. Add a class to the diagram
3. Right click on the class in the diagram and select Stereotypes...
4. In the dialog check the box called simplegrlprofile::Goal and click ok.

B3: How to create the structure for a (metamodel) profile in TAU

This section describes how to create the necessary structure for metamodel profile in TAU. Generally a metamodel profile contains a large number of elements and can be implemented a number of ways. For this reason this section will outline how a metamodel profile is created and highlight the critical aspects of the GRL Profile. It is assumed that the reader will have access to the already completed GRL Profile created and that reader will use this as a reference when going through this section of the document.

Download and Install the TAU FIDebugger

The FIDebugger is a part of the TAU G2 SDK. It will be needed to read the Global Unique Identifiers (GUID)⁵ of the GRL metamodel elements in the GRL Profile

1. Download the TAU G2 SDK from the telelogic website
2. Follow the readme and install the FIDebugger

⁵ Refer to the TAU help documentation for a full description of GUID's in TAU

Create the Profile Directory Structure:

1. Follow the instruction in B1 - Create the Profile Directory Structure (but use the name GRLProfile instead of simplegrlprofile)
2. Change the GRLProfile.mod to file to one that looks like the one below

```
[GRLProfile]
"scope"="PROJECT"
"version"="1.0"
"longname"="GRL Profile"
"description"="GRL Profile ability"
"product"="elvis"
[GRLProfile/Bin]
"listBin"=""
[GRLProfile/Script]
"listScript"="load.tcl"
[GRLProfile/Etc]
"listEtc"="urn:u2:addins/GRLProfile/etc/GRLProfile.u2"
```

Note: the main difference is the tcl script

Create the Profile Project in TAU:

1. Launch TAU G2, and create a UML Modleing Project named simplegrlprofile in the etc directory created above
2. Expand the Model, and right click on the package and sleect Stereotypes...
3. In the Dialog check TTDPrefinedStereotypes:profile

Create Sub Packages for the metamodel profile:

1. Create the following sub packages in the main GRL Profile package:
 - Models – this package will contain metaclasses describing the GRL Model and its elements
 - Diagrams – this package will contain metaclasses describing the actual GRL diagram and its elements
 - GRLLinks – this package will contain the metaclasses representing the GRL links
 - GRL Elements – this package will contain the metaclasses representing the actual GRL elements
2. For each of the listed packages (and the main package) add the following stereotypes
 - TTDExentsionManagement::browserModel
 - TTDExtensionManagement::propertyModel
 - TTDPrefinedStereotypes:metamodel
 - TTDPrefinedStereotypes::profile

Note: Use the already Created GRL profile as reference to see what the associated values of the the stereotypes in the properties of each package should be

Add the metamodel classes representing the core UML structure:

In TAU when creating a metamodel customizing UML, the basic UML structure that is desired has to be added to the new metamodel. In the GRL Profile, most of the non-package elements in the main package represent the core UML structure needed. Of these elements the most important is the rootElement. This metaclass links the new GRL metamodel in to TAU. It is important to examine the properties of the all the elements in already created GRL profile. The best way to view all of an elements properties is to:

1. Right click on the element and select Properties
2. In the Edit Properties view, choose the Filter, All Properties
3. This will list all the fields associated with each of the stereotypes applied to each of the metaclasses (the values of these fields are very important)

Note: There are two stereotypes that are applied to almost all the elements in the profile TTDExensionManagement::browserModel and TTDExensionManagement::propertyModel

Add the metamodel classes represent the GRL customizations

The metaclasses in the packages GRLLinks and GRL Elements contain the actual GRL specific elements. The packages Model and Diagrams contain the metaclasses that hook the GRL elements in the UML metamodel of TAU. For full detail refer to the class diagrams in the GRL Profile.

Create the TCL script for the Profile

This portion discusses how to create the script that launches the GRL profile⁶.

1. Go to the menu Tools → Debugger (this launches the FIDebugger dialog)
2. In the Model View, select the main package of the profile GRLProfile
3. The Debugger will display all of the info about the package
4. In the Debugger dialog copy the GUID of the package
5. Now Save and Close TAU
6. Using Windows Explorer, Naviagate to <TAU Install
Dir>\addins\GRLProfile\script\
7. Create a text file called load.tcl that looks like

```

package require commands
#package require dialogs
# Returns the session of the active project. Should be used whenever
the session
# is needed.
proc GetActiveSession {} {
    set activeProject [std::GetActiveProject]
    return [lindex [std::GetModels -kind U2 -project $activeProject] 0]
}

set ProfilePath [file join [std::GetInstallationDirectory]
addins/GRLProfile/etc/GRLProfile/GRLProfile.u2]
output "Loading GRLProfile ..."
u2::LoadLibrary $ProfilePath

set GRLMetaModel [u2::FindByGuid [GetActiveSession]
"QlhHIIuL3KVLJ89hjVuJ76RI"]
if {$GRLMetaModel != 0} {
    RegisterMetaModel $GRLMetaModel
    u2::SelectMetaModel $GRLMetaModel
}

output "Done.\n"

```

⁶For a full description of TCL scripts refer to Help documentation in TAU

B4: How to use a (metamodel) profile in TAU

This section describes how to activate and use the GRLProfile metamodel

Put the GRL Metamodel in the Add-ins dir (if its not there already)

1. Make sure that TAU is closed
2. Copy the GRLProfile directory in the <TAU Install dir>/addins
3. Launch TaU

Create a UML Modeling Project and activate the GRLProfile metamodel

1. In TAU create a new UML Modeling Project call it Sample
2. Go to the menu Tools→Customize
3. In the Customize dialog, select the Add-ins tab
4. Check the box labeled GRLProfile and click close
5. [GRLProfile] should appear next to the name of the Modeling Project in the Model view

Using the GRLProfile metamodel

1. Expand the Modeling Project in the Model View
2. Select the main package
3. Right click on the main package and go to New→ GRL Diagram
4. A GRL Diagram should be created with a customized palette

B5: Reference (metamodel) Profiles in Tau

The core UML metamodel in TAU is called the TTDMetamodel. A description of this metamodel can be found in the Library section of any Tau Modeling Project. A full description of this metamodel profile is in the TAU SDK. A good overall understanding of this metamodel is required to use metamodel Profiles in TAU

Another Profile that could be used as a reference is the DataModeler metamodel Profile. The DataModeler profile has customized diagrams and elements used to express Databases models. For more information refer to the documentation inside this profile.

Reference:

[1] *UML Infrastructure Specification*:

http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML

[2] GRL Website:

<http://www.cs.toronto.edu/km/GRL/>

[3] *URN Website*:

<http://www.usecasemaps.org/urn/index.shtml>

[4] *Introduction to User Requirement Notation: Learning by Example*, Daniel Amyot

[5] How to Create a Profile in TAU G2: **Extending Tau Using its Add-In Functionality**, Greg Gorman