The UDM Interpreter Generator for Generic Modeling Environment

Tihamér Levendovszky
05/14/2002
(updated 08/30/2004)

0. Table of Contents

1. Introduction

The GME interpretation process needs to access to MGA objects of a created model. An approach can be to use the **B**uilder **O**bject **N**etwork [1] shipped with GME to traverse models, change MGA objects in the models and generate the desired output. Similar to BON, UDM [2] offers facilities for accessing MGA via UDM GME backend, and makes the current project accessible by raw COM interpreter interface. To ease the development a generator has been developed which creates a Visual Studio 6.0 project with the necessary capabilities to provide a programming environment for using UDM. The objective of this document is to describe the basic steps to write a UDM-based interpreter for GME assuming medium-level familiarity with UDM programming (it involves STL) and GME.

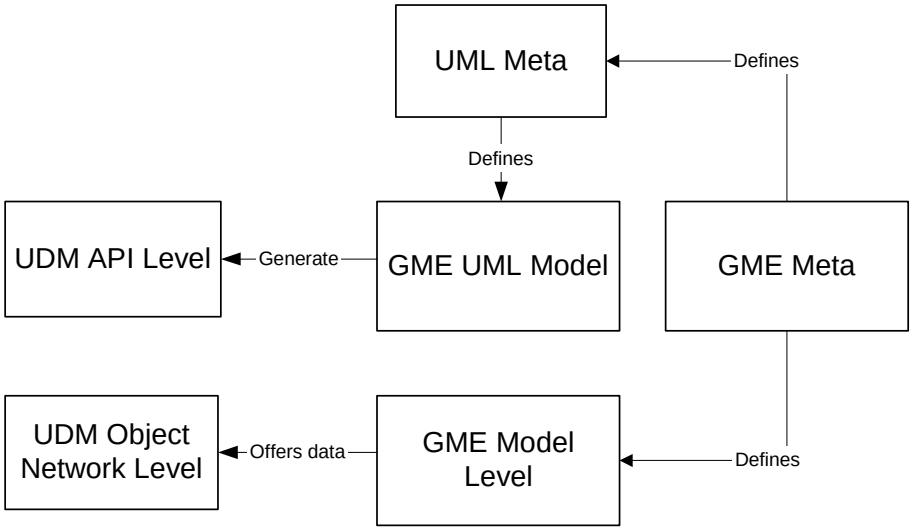

Figure 1. Correspondence between GME and UDM

Figure 1 shows the how GME and UDM fit together. In GME there must be a metamodel which defines the modeling environment, and a model that is drawn using this environment defined by the meta.
Separately from that UDM also uses this environment to express its meta: a GME UML paradigm (meta) defines a UML class diagram environment to specify UDM meta with this class diagram. In this configuration the UDM meta is a GME model (class diagram).
To create UDM meta from a GME meta we need to convert the GME meta to the GME UML class diagram paradigm.

2. Generating the Meta

Having a GME metamodel is created to develop the modeling environment we also have to create a meta for the UDM environment. This process has not been automated yet, so UDM meta must be created manually.

2.1 Meta Creation Process Overview

The UML class diagram can be generated from the GME meta using the GME2UML interpreter. This is available with the GReAT installation. Follow the instructions for the GME2UML interpreter to generate the UML class diagram.

The UML class diagram should then be interpreted using the UML2XML interpreter that comes with UDM. In fact this interpretation process creates a DOM backend of the metamodel without the corresponding DTD file. Udm.exe must then be run on this XML file. Udm.exe creates *.h and a *.cpp file along with the DTD file. These files are the code to set up a static memory backend for the meta-information.

3. Using the Generator

The Generator is a Windows Wizard application with a well-known user interface. To use it you need Windows 98 or later and Visual Studio 6.0.

3.1 Meta loading and caching

The configuration options for meta loading and caching are presented in Figure 2.

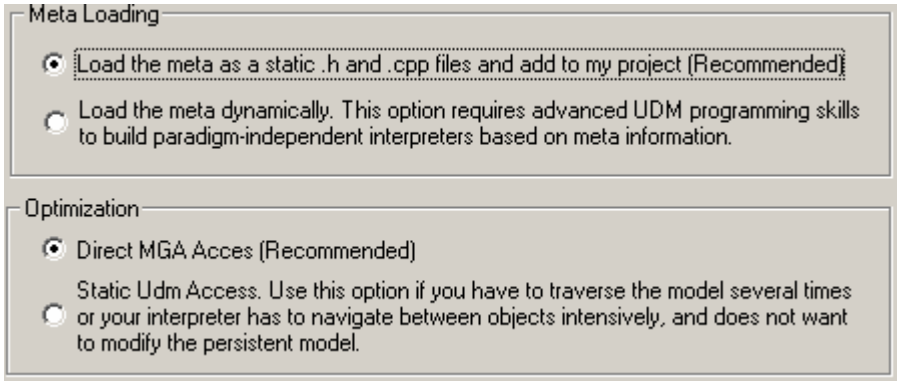

Figure 2. Meta loading and caching configurations

3.1.1 Dynamic vs. Static Meta

Static meta must be compiled to the interpreter. This is the perfect solution for supporting only one meta (paradigm), and most of the time it is enough.

Dynamic meta is based on the DOM representation generated by the UML class diagram interpreter. When one wants to write a GME interpreter, which supports multiple paradigms, and the meta is another input data along with the model, this is the appropriate option. Because the meta is not available at programming time, it is uncomfortable to traverse the model and writing this code needs advanced UDM programming skills and familiarity with UDM insights.

3.1.2 Optimization

For optimization purposes the generator offers creating a memory backend (static data network) and copies the model to this backend. That means that every model element will be read from MGA, and in case of huge models this takes significant amount of time. On the other hand if your interpreter traverses the models several time and reads it more than once, it is worth using this facility. Currently writing data back to MGA is not supported, modifications in the cache will not be saved to GME.

3.2 Configuring UDM Development Environment and meta paths

3.2.1 UDM Environment Settings

Enter the path of the UDM installation. The default path taken from the environment is already inserted, but you may change it to another installation folder where you have installed UDM.

Figure 3. UDM Environment settings

3.2.2 Meta settings

Based on the selection made in the previous step there are two different options here.

Figure 4. Code-based meta

If we chose static backend (Figure 4) we must specify the path for the .cpp and .h files. These will be copied to the project folder and will be added to the Visual C++ project.

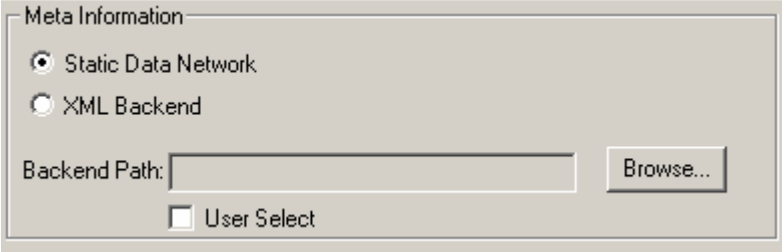

Figure 5. Dynamically loaded meta

If the meta is loaded dynamically (Figure 5.) we can chose a memory backend or an XML file. This path will be stored in the *config* global variable in the interpreter code. If **User Select** is selected, a code snippet is generated and the interpreter will ask the user to select the meta at run-time.

3.3 Component Settings

The component settings (Figure 6.) are required by the GME interpreter interface. We must provide a name for the component and specify the paradigm name we want to interpret. An interpreter can be registered for all paradigms.

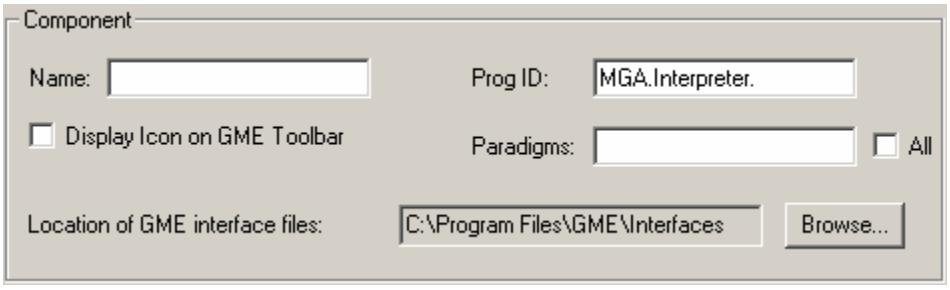

Figure 6. Component Configuration

An icon (included in the project) can be displayed on GME toolbar after registering the interpreter.

5. The Generated Code

The generated code contains two main configuration files, one for UDM (UdmConfig.h), and one for GME (ComponentConfig.h). Most of the settings specified in the generator can be modified in these files.
The UdmApp.cpp file contains the main entry point of the UDM-based interpreter. It is passed an open backend pointer that can be memory (if caching is enabled) or a GME

backend pointer. The *focusObject* is the currently open model window. If we display icon on GME toolbar this can be null object, otherwise not. The *selectedObjects* are the currently selected GME objects that also can be null objects if no objects selected. The framework caches all the exceptions to protect GME from unhandled exception crashes caused by this in-process COM component interpreter.

6. References

[1] *ISIS*: **GME User's Manual,** ISIS Vanderbilt University, part of GME distribution
[2] *Bakay* A.: **The UDM Framework**, ISIS Vanderbilt University, 2001 December