Gmodel

Semantic Modelling Platform

Version 1.0

Product Description

(as of 1st of July 2018)

Sofismo AG
http://www.sofismo.ch

Niederlenz, Switzerland
Tel. +41 76 317 66 74

1 Overview

Gmodel is a metalanguage and metadata repository that has been designed from the ground up to enable specification and instantiation of modelling languages. Although a number of metalanguages can be used for this purpose, most provide no or only limited support for modular specifications of sets of complementary modelling languages. Gmodel addresses modularity and extensibility as primary concerns, and is based on a small number of language elements that have their origin in model theory and denotational semantics. Gmodel offers support for unlimited multi-level instantiation in the simplest possible way. Therefore, any metalanguage emulated in Gmodel can optionally be equipped with this functionality.

2 Gmodel Open Source Edition

The Gmodel Open Source Edition is a repository-based platform for model engineering that is delivered as a set of Eclipse plug-ins that can be deployed within the Eclipse Modeling Tools platform (http://www.eclipse.org/modeling/).

2.1 Features

A. Semantic domains
Gmodel separates the concern of modelling from the concern of naming. Right down to the core Gmodel functionality is expressed in semantic identities, and these identities can be referenced from as many representations (models) as needed. Users can create semantic domains that consist of a set of semantic identities, and users can use composition to modularise semantic domains.

B. Multi-level instantiation
The most powerful feature of Gmodel instantiation is the ability to decorate any Gmodel artefact with instantiation semantics (or concretisation semantics) relating to representations of less abstract (or more concrete) sets, such that the artefact becomes instantiable. The instantiation semantics available in the Gmodel kernel are specified via the variables for cardinalities, navigability, and artefact containment. Gmodel is not limited to the four-layered metamodel architecture. Users can continue recursively with instantiation and decoration over any number of levels.

C. Extensible metametamodel
Users are able to extend the metametamodel of Gmodel. This opens up new approaches with respect to model-based interoperability, since types and therefore interoperability patterns can be encoded in Gmodel to any level of complexity.

D. Variant management
The extensibility of the metametamodel in combination with multi-level instantiation enables users to create precise models of the commonalities and variabilities found in highly configurable product lines.

E. Scope management
Gmodel has an explicit feature for scope management that is universally available within all modelling languages expressed in Gmodel. This gives designers of modelling languages and system architects an unprecedented amount of control over the artefacts that language users can instantiate. A model artefact may not reference any element in other model artefacts unless these artefacts have been declared to be visible from the first artefact.

F. Referential integrity
The Gmodel repository provides referential and transactional integrity relating to model artefacts.

G. High-performance persistence via relational database
By using relational database technology Gmodel is able to scale to large numbers of models in the same way that operational relational databases are able to deal with millions and billions of data items.

H. Non-graph content management
Every Gmodel artefact is capable of holding a “payload” of unstructured data. This feature allows Gmodel to be used as a universal Enterprise Content Management system.

I. Java API
The entire functionality of Gmodel is exposed in a simple Java API. In contrast to many other modelling technologies, Gmodel makes no assumption about the implementation and legacy technologies that modellers are going to drive from their model artefacts. The Gmodel kernel is highly portable. It makes use of the Java programming language to bootstrap the nine kernel concepts – but without exposing the Java type system in the API.

J. Model visualisation (graphs + visual forms)
At this point in time Gmodel provides two complementary graphical surface notations for visualising Gmodel artefacts.

K. Model-driven generation
Out of the box, Gmodel includes integration with the Eclipse integrated development environment, and with the openArchitectureWare Xpand template/transformation engine, putting text or code generation at the user's finger-tips.

2.2 Current Road Map

The current plan for developing further Gmodel Open Source Edition functionality includes the following intentions (feature details, priorities, and delivery dates may change depending on input and feedback from the Gmodel user community):

A. Full text repository search
[Q3 2010]

B. Persistence of preferred diagram layout
[Q3 2010]

C. Artefact state machines and event handling
[not scheduled, yet]

D. Bi-directional dynamic bridge to EMF Ecore
   [not scheduled, yet]

3 Gmodel Semantic Extensions Edition

The Gmodel Semantic Extensions Edition is intended as an advanced repository-based platform for model engineering with a web based user interface for semantic modelling.

3.1 Current Road Map

The current plan for developing Gmodel Semantic Extensions Edition functionality includes the following intentions (feature details, priorities, and delivery dates may change depending on input and feedback from the Gmodel user community):

A. Multi-language and multi-jargon capability
   [Q4 2010]

B. Time conscious artefacts
   [Q4 2010]

C. Optional hosting of repository content
   Q4 2010]

D. Web-based generic model editor (incremental delivery of more and more advanced features)
   [Q2 2011]

E. Change-impact analysis functionality
   [Q2 2011]

F. Role based access control
   [Q2 2011]

G. Task management
   [Q2 2011]

H. Model refactoring functionality (incremental delivery)
   [Q4 2011]

4 Gmodel Technology Glossary

4.1 Set

The generic term to refer to any concept that is expressed in Gmodel is the **set** – in the mathematical sense. Gmodel also makes use of the basic set theoretic concepts of **sub set** and **super set**.

4.2 Ordered Pair

The simplest set is the **ordered pair** – in the mathematical sense, which in Gmodel is defined as consisting of an element **e** and a so-called meta-element **m**. Thus in mathematical notation an ordered pair $\mathbf{o} = \{\mathbf{e} , \mathbf{m}\}$.

4.3 Ordered Sets

Ordered pairs are used in Gmodel to encode **ordered sets** – in the mathematical sense. In an ordered set $\mathbf{s} = \{\mathbf{e}_1, \mathbf{e}_2, \dots \mathbf{e}_n\}$ an “order” relation **<** (is-smaller-than) is defined, and the relation $\mathbf{e}_i < \mathbf{e}_j$ is true for all $\mathbf{i} < \mathbf{j}$.

4.4 Graphs

A **graph** is a set of vertices and a set of edges, where an edge is a set of two vertices. In mathematical notation a graph $G = \{\mathbf{v}_1, \mathbf{v}_2, \dots \mathbf{v}_n\} \cup \{\mathbf{e}_1, \mathbf{e}_2, \dots \mathbf{e}_m\} = \{\mathbf{v}_1, \mathbf{v}_2, \dots \mathbf{v}_n\} \cup \{\{\mathbf{v}_{i1}, \mathbf{v}_{j1}\}, \dots \{\mathbf{v}_{im}, \mathbf{v}_{jm}\}\}$.

4.5 Artefact

In Gmodel all **artefacts** (sometimes also called *work products*, and often represented as *files* or *database records* in computer systems) are encoded as graphs. More precisely, a Gmodel artefact consists of a set of vertices and several complementary ordered sets of links:

A. **Edges** – links between two sets with a dedicated edge end for each connected set. Amongst other things, when modelling information systems, Gmodel edges can be used to represent *relationships* in entity relationship models, and they can be used to represent *associations* in object oriented models.

B. **Super set references** – directed links from a sub set to a super set. Amongst other things, when modelling information systems, Gmodel super set references can be used to represent *generalisation/specialisation* hierarchies in extended entity relationship models or in object oriented models. In product line engineering, super set references can be used to represent *variants* of artefacts and hierarchies of variants of artefacts.

   Note 1: The Sofismo Technology License makes use of the super set reference concept to precisely differentiate between artefact *instantiation* and the creation of artefact *variants*. The relationship between an artefact and an instance of the same artefact roughly corresponds to the relationship between a *class* and an object in *object* orientation, whereas a super set reference to an artefact roughly corresponds to a *specialisation/generalisation* link (inheritance) in object orientation. Taken together, clauses 1.6., 3., and 5. in the Sofismo Technology License mean that you (the licensee) are granted the right to instantiate artefacts, and are also allowed to create an unlimited number of variants of the instances that you have created, but you are not allowed to create variants of artefacts that are part of the Original Code (the Gmodel Semantic Extensions product). The situation is comparable to the rights that you are granted by the manufacturer of a compiler – you can use the compiler to create (instantiate) software artefacts that you own (and can sell), but you have not been granted the right to create variants of the compiler itself.

Note 2: One of the unique features of Gmodel is the capability of unlimited multi-level instantiation. This means that Gmodel allows you to create artefacts that are instantiable, and the resulting instances can again be instantiated as needed and so on. If you are intending to use Gmodel as a run-time technology as part of your software product, your customers need a Limited (run-time) license for Gmodel. To protect your intellectual property (to prevent your customers from creating – and selling – variants of your artefacts), you can use a licensing approach with your customers that is similar to the approach used in the Sofismo Technology License.

C. **Visibilities** – directed links from one sub graph to another sub graph. Amongst other things, when modelling information systems, Gmodel visibilities can be used to represent the *scope definitions* that are found in *include* and *import* statements in some information modelling and programming languages.

D. **Edge traces** – directed links from one edge to another edge. Amongst other things, when modelling hardware or software systems that consist of *nested components*, Gmodel edge traces can be used to represent the links between the internal elements connected to a port and the external elements connected to a *port*.

4.6 Instantiation

A concept always represents a set of things. In Gmodel these sets are called *artefacts*. **Instances** are those artefacts that feel like unique items from the perspective of a specific role that can easily be distinguished from each other.

Example 1: In a family that owns a single car, the family members will talk about using the "car" - and it is clear that it is "the" car. The family might even sell the car and buy a new one, and they will continue to talk about the "car" - the car is perceived and treated as an instance from the perspective of the family. This example also nicely illustrates that the instance concept is independent of the concept of identity. The identity of the new car differs from the old car!

Example 2: Now consider a family that owns two cars. The cars are likely going to be referred to as the "Golf" and the "Peugeot", and from the perspective of this family the "Golf" and the "Peugeot" are instances.

Example 3: Next consider a car sales person who is talking about next year's Golf, and then about the specific X Edition Golf and the Y Edition Golf (both this year's models) sitting in the show room. Here, on the one hand the sales person is referring to the latest Golf concept as it has been defined from the perspective of the marketing and engineering team at VW, and on the other hand he is referring to the X Edition Golf and the Y Edition Golf models from the perspective of the specific dealer. As soon as one person is referencing concepts from different perspectives it gets very tricky to follow the conversation without misinterpretation (= non-alignment of desired intent and semantics).

Mathematically speaking, **instantiation** is the act of assigning *values* to *variables*, such that the result is perceived as a unique item from the perspective of a particular role. The role in question may then decorate the instance with further variables (usually this is called *meta modelling*) to enable a "downstream" role to assign further values (usually this is called *modelling*) that lead to a result that is perceived as a unique item at the lower level of abstraction corresponding to the perspective of the downstream role.

4.7 Semantic Modelling

Gmodel makes use of the theory of **denotational semantics**, in particular the concepts of *semantic domain* and *semantic identity*. Gmodel clearly distinguishes between semantic domains and models. The former simply contain sets of semantic identities, whereas the latter contain representations of semantic identities from the view point of a particular actor.