

The Generic Eclipse Modeling System (GEMS):

A Proposal to the Generative Modeling Technologies (GMT) Project

By Jules White

Version 1.0

Introduction

The goal of the Generic Eclipse Modeling System (GEMS) is to bridge the gap between the communities experienced with visual metamodeling tools, such as the Generic Modeling Environment (GME), and those built around the Eclipse modeling technologies, such as the Eclipse Modeling Framework (EMF) and Graphical Modeling Framework (GMF). GEMS is being developed by the Distributed Object Computing (DOC) Group at Vanderbilt University's Institute for Software Integrated Systems (ISIS) and other collaborators, such as Siemens Corporate Technology. GEMS is an open project and encourages developers to extend, enhance, and use its tools. GEMS has been developed in conjunction with research work done in collaboration with Siemens, IBM, and PrismTech. GEMS is an open source project, based on the Eclipse License.

Goals

GEMS is a project aimed at bringing the deep experience in existing visual metamodeling techniques and tools, such as GME, gained by the DOC group and other researchers and developers, to the Eclipse platform and bridge the communities. GEMS is designed to provide a tool for rapidly creating Eclipse modeling tools from a visual metamodel specification with little or no coding and allow developers and users to focus on the code generation and analysis aspects of a graphical modeling tools rather than the intricate user interface code that is traditionally required for graphical modeling tools. Our intention is to bring the extensive experience of existing graphical metamodeling tool users in formal specification, analysis, simulation, modeling, and other techniques to the Eclipse platform and provide a point of integration between the two communities. The project hopes to provide a bridge in terms of use as well as interoperability between the Eclipse modeling tools and existing graphical metamodeling toolsets in the form of bi-directional model and metamodel exchange.

A key research aspect of the work will focus on enabling complex analysis, simulations, and constraint satisfaction. In particular, GEMS is specifically targeted towards addressing the scalability challenges of manual modeling approaches. A facility called Model Intelligence Guides (MIGs) is included within GEMS to simplify the integration of intelligent modeling assistance mechanisms, such as applying batch changes to a model to make it meet a set of global constraints.

Relationship with other Eclipse Tools: The graphical modeling environment, provided by GEMS, is based on EMF, Draw2D, and GEF. GEMS' goal is to be a tool that allows domain experts to quickly create graphical editors using EMF, Draw2D, and GMF. We are currently transitioning our code generators to produce GMF-based code rather than bare Draw2D and EMF code. GEMS is a tool that allows developers to quickly and simply tie together multiple Eclipse modeling technologies with little or no Java or XML coding experience. We will continue to evolve GEMS to expose the best features of the Eclipse modeling plug-ins, such as EMF and GMF, that it is based on, while focusing on making the process of creating a DSML simple. Again, the goal of GEMS is to allow domain experts, not just Java/Eclipse developers, to create a DSML that leverages the powerful modeling frameworks for Eclipse, such as GMF and EMF, and create complex analysis, optimization, and code generation facilities.

Relationship with the Generic Modeling Environment: GEMS provides a meta-programmable modeling environment similar to GME for Eclipse. Currently, plug-ins exist for GEMS to do basic importation of GME metamodels and models into GEMS. Our goal with respect to GME is to extend GEMS to provide true bi-directional interoperability between the modeling tools.

Relationship with Other Graphical Metamodeling Tools: We encourage others to develop extensions to GEMS to interoperate with other graphical metamodeling or CASE tools, such as MetaEdit.

Context

Modeling scalability challenges for large systems. Domain-Specific Modeling Languages (DSMLs) [1], Model-Driven Engineering (MDE) [2], Model-Driven Architectures (MDA) [3], and Model Driven Development (MDD) [4] are all promising forms of system development that combine high-level visual abstractions, specific to a domain, with constraint checking and code-generation to simplify the development of a large class of system [5]. As model-based tools and methodologies have developed, however, it has become clear that there are many domains where the models are so large and the domain constraints so intricate that it is extremely difficult for a modeler to manually produce a correct or high quality model. In these domains, modeling tools that merely provide solution-correctness checking via constraints or a visual representation of the domain, provide few real benefits over a model-less approach. The true complexity in these domains is their large model sizes and the combinatorial nature of their constraints-not code construction per se. For example, specifying the deployment of software components to Electronic Control Units (ECUs, the automotive equivalent of a CPU), in a car, while observing configuration and resource constraints, even when only a few tens of model entities are present, can easily generate solution spaces with millions or more possible deployments and few correct ones. For these large modeling problems, it is impractical, if not impossible; to create a complete and valid model manually.

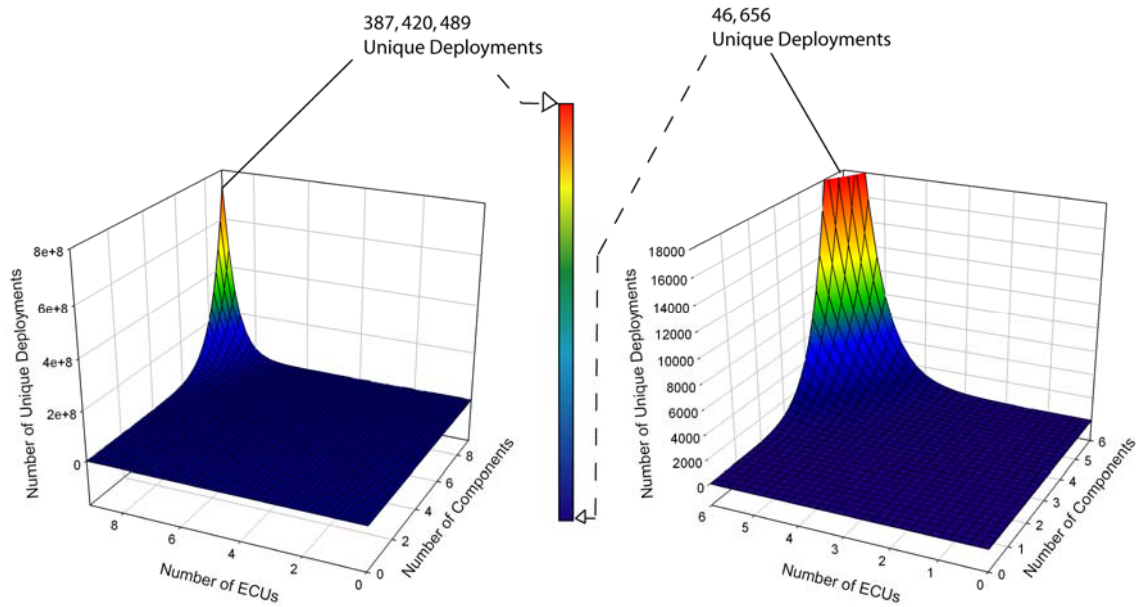


Figure 1, Comparing the Number of Unique Deployments for Two Model Sizes

Consider a group of 10 components that need to be deployed to one of 10 ECUs within a car. There are $10^{10} = 10 \text{ billion unique deployments}$ that could be tried. Part of the complexity of these domains is how quickly the solution spaces grow as the number of model elements increases. To visualize the speed at which the solution spaces grows for our automotive example, examine Figure 1. With 9 components and 9 ECUs we have a total of 387,420,489 unique deployments. It appears from the first figure that the solution space size is relatively flat when there are less than 6 components and 6 nodes. If you examine the second graph, however, you will see that the solution space is actually not flat at all from 0-6 components/nodes but only appears flat when scaled in comparison to 9 components/nodes. Clearly, any approach to finding a deployment that observes the deployment constraints must be efficient and employ a form of pruning to reduce the time taken to search the solution space. A manual approach may work for a model with 5 or so elements, but as can be seen from Figure 1, the solution space can increase rapidly as the number of elements grows.

Typically, each component in an automobile will have multiple constraints governing its placement. The Anti-lock Braking System (ABS) will need to be hosted by a controller at least a certain distance from the perimeter of the car. Furthermore, the ABS will have requirements governing the CPU, memory, and bus bandwidth available on its host. When these constraints are considered for all the components, it becomes difficult for a modeler to manually produce a correct solution. This example only has 10 components and 10 control units. Real automotive models typically contain 80 or more control units and hundreds of components. In models of this scale, manual approaches simply cannot handle the large numbers of possibilities and the complexity of the constraints.

GEMS is designed to allow developers to rapidly create a graphical modeling tool with no coding and immediately begin addressing the challenging aspects of these types of complex domains. GEMS provides extensive support for integrating intelligent

mechanisms into a modeling tool to provide visual modeling queues, constraint-compliant batch processing, simulation, and analysis.

Approach

Currently, GEMS is developed by a group of developers led by the DOC group. Our collaborators include Siemens Corporate Technology, IBM, PrismTech, and others. The project is open and encourages others to join its development and user community.

GEMS provides extensive support for rapidly creating graphical modeling tools for Eclipse using a visual modeling environment. The true research focus of GEMS, however, is in addressing the scalability problems of manual modeling approaches in complex and large domains. GEMS' approach to this problem is presented below.

Solution approach → Modeling tool and constraint solver integration. To address the challenges of modeling large and complex domains, methods are needed to reduce the cost of integrating constraint solvers with modeling tools. GEMS provides a mechanism called Model Intelligence Guides (MIGs) to substantially reduce the cost of integrating a constraint solver and reduce the complexity of modeling hard domains by:

1. **respecting domain-specific concepts** from the modeling tool and providing a flexible mechanism for specifying solvers using domain notations. Modeling tool users are able to specify constraints in a language or notation that mirrors the domain and makes mapping requirements to the model easier.
2. **leading modelers towards solutions that are considered optimal** or good based on quality metrics from the domain. MIGs allows solvers to be used to iterate through multiple valid solutions and suggest only those considered most optimal. Modelers can plug-in custom formulas for measuring optimality in the target domain and the tool and MIGS can present multiple suggestions based on various types of optimization.
3. **automating tedious and complex modeling tasks**, such as solving for and assigning values for global constraints, performing repetitive localized decisions, or providing feedback to a modeler to suggest valid modeling decisions.
4. **accommodating long-running analyses** for problem instances that cannot be solved on-line.
5. **providing debugging information or suggest ways to make the model tractable.** In a domain that requires constraint solver assistance, it is unlikely that a manual approach will be able to find the source of intractability in a model. Thus, it is crucial that MIGs provides feedback to the modeler to facilitate debugging.

With the constraint-solver integrated modeling environment, provided by GEMS, a user goes through an iterative process of specifying portions of a model, adding or refining existing domain constraints, debugging constraints that are not tractable, and using the constraint solver to automate model construction and optimization. Figure 3 illustrates the modeling processing with an integrated constraints solver.

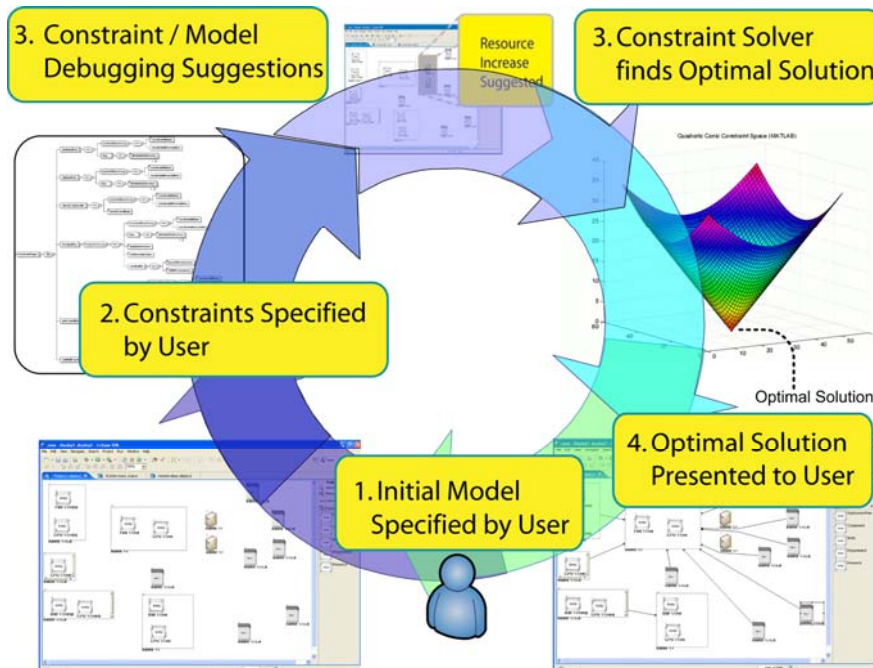


Figure 3, A Modeling Cycle with Constraint Solver Integration

MIGs can guide the user in making localized decisions by highlighting optimal or valid model modifications and decrease user effort and modeling time. MIGs can also be used to complete a batch process of model changes, such as connecting all components to a valid host, while respecting global constraints. Batch processes performed by MIGs arrive at higher quality solutions in less time than a manual approach. Finally, in situations where the partial model specified by the user, such as a listing of components and nodes they must be deployed to, cannot satisfy the domain constraints, e.g. the modeled nodes do not have sufficient resources to host the components, MIGs can suggest ways of bringing the model to a state that satisfies the domain constraints. MIGs might, for example, indicate that a particular node's available memory must be increased to satisfy a failed resource requirement.

GEMS Features

GEMS provides an extensive feature set for rapidly building graphical modeling tools including:

- A graphical language for metamodel specification that can capture
 - domain entities
 - attributes of entities
 - inheritance relationships
 - connection and containment relationships between entities
 - distinct modeling views
 - graphical information, such as connection styles, colors, and fonts
 - constraints

- GEMS provides extensive support for intelligent modeling guidance including
 - Automatic solving of inference-based constraints for connection and containment
 - Applying batch changes to a model that are guided by a set of global constraints
 - Visually suggesting connection endpoints and element parents that conform to the specified constraints
 - Support for simulations
 - Mechanisms for integrating constraint solvers
 - Mechanisms for creating re-usable templated constraint solvers
 - Mechanisms for discovering why a model cannot meet a set of global domain constraints
 - Graphical mechanisms for suggesting optimal modeling decisions derived from a constraint solver
- A code generation framework, which does not require any coding or XML editing, for transforming a GEMS metamodel into a working Draw2D/GEF Eclipse plug-in for editing instances of the language
 - Ecore Model
 - EMF Classes
 - GEF/Draw2D Figures
 - GEF/Draw2D Edit Parts
 - Other GEF/Draw2D Infrastructure for creating the palette, etc.
 - Plugin descriptor for the tool
 - Build descriptor for the project
 - Classpath descriptor for the project
- The visual appearance of the generated modeling tool can be customized by creating CSS stylesheets to modify the icons, colors, fonts, connection styles, and other visual attributes of the modeling entities
- The views available to a modeler can be customized through a mechanism similar to CSS stylesheets
- The generated graphical modeling plug-ins, created by GEMS, support extensive external customization through extension points for
 - Adding code generators and transformation, such as Open Architecture Ware, Java Emitter Templates, and Atlas Transformation Language
 - Model pre and post processing
 - Model event listeners
 - Triggers for invoking actions (similar to database triggers)
 - Constraint languages
 - Menus
 - Palette customizers
 - Intelligent modeling guides
 - Model serializers
- GEMS provides built-in support for constraints written in Java, OCL, and Prolog
 - Constraints can also be used as triggers for invoking actions
- Models and constraint solvers can be accessed remotely using a built-in CORBA server

GEMS supports many other features in addition to these. New features and functionality are continually being added to GEMS as it develops.

Target audience / End users

GEMS is currently used in various projects at Siemens, relating to AUTOSAR, and the Distributed Object Computing Group at Vanderbilt University. GEMS' target audience is developers and researchers, such as those experienced with GME, who need to focus on large and complex modeling domains where analysis, simulation, and code-generation are the key challenges. GEMS is also targeted towards users who want to create graphical modeling tools for Eclipse from a visual metamodel specification without any XML/Draw2D/GEF coding. As the project develops, our goal is to focus on continuing to improve the mechanisms for providing modeling guidance and the visual stylability of the generated tools.

Bibliographical references

1. Ledeczi A, Bakay A, Maroti M, Volgysei P, Nordstrom G, Sprinkle J, Karsai G (2001) Composing Domain-Specific Design Environments. IEEE Computer, November.
2. Kent S, (2002) Model Driven Engineering. In: Proc. Integrated Formal Methods: Third International Conference, Turku, Finland, May.
3. Kleppe A, Bast W, Warmer JB, (2003) The Model Driven Architecture: Practice and Promise, Addison-Wesley Professional, NY, USA.
4. Selic B, (2003) The Pragmatics of Model-Driven Development. IEEE Software, IEEE Computer Society Press, Los Alamitos, CA, USA.
5. Sztipanovits J, Karsai G, (1997) Model-integrated computing, IEE Computer, IEEE Computer Society Press, Los Alamitos, CA, USA.
6. Ledeczi A (2001) The Generic Modeling Environment. In: Proc. Workshop on Intelligent Signal Processing, Budapest, Hungary.
7. Van Hentenryck P, Saraswat V, (1996) Strategic directions in constraint programming. ACM Computing Surveys, 28, 4, ACM Press, NY, USA.