Domain Specific Modelling Language Implementations in Tools

R. RAMESH[1], Dr. V. PRASSANAVEKATESAN[2]

[1]Research Scholar, Department of Computer Science and Engineering, Pondicherry University, Pondicherry, India.
[2]Associate Professor, Department of Banking Technology, Pondicherry University, Pondicherry, India.

***Abstract*-** DSM improves the quality; increase the productivity and hiding the complexity. It is indispensable to explain the Domain-Specific Modeling Language and generator's working methods. Development of Domain Specific Modeling simplifies and speeds up the programming into many folds. DSM Tools are very effective in developing models and in code generations. This paper explains about DSML concepts and DSML implementation among the tools. In general modeling technique is applied for specifying required functionality and creating documentation for better understanding of the system. Features and advantages of the tools are also explained in detail. Architecture of each tools are discussed in detail to understand the DSML Implementation in the tool. The necessary conditions for implementing DSML are discussed.

Index Terms- Domain, Model, MetaCase, Generic Modeling Environment, Domain Specific Modeling, Domain Specific Language.

I. INTRODUCTION

In this paper we plan to discus in short about modeling tools like MetaEdit+, GME and Sequencer. MetaEdit+ provides screen options to enter associated rules, properties and symbols. The Entered rules and facts are stored as a metamodel in a repository and it permits modifications.

UML solved it first in terms of domain with very little tool support. UML relate model development directly to implemented domain, i.e. it visualizes the code. The domain solution must be related to the UML1 models. The developer should fill up the method bodies in the largest part of the implementation. Thus the developer has to work out the problem twice i.e. in domain terms and in code terms.

GME (Generic Modeling Environment), is design model integrated program and tool. The GME is a Windows-oriented, domain specific, model-integrated program integrate tool in creating and modifying domain specific, multi-aspect models of large-scale systems. The fast development of DSML2 tools is a crucial issue in model driven development. To raise the level of abstraction in model-driven development, both the modeling language and the generator need to be domain-specific, i.e. restricted to develop only certain kinds of applications. While it is trivial that we can't have only one code generator for all software, it is surprising to many that this applies for modeling languages tool

To describe a modeling language involves three aspects: the domain concepts, the notation used to represent in graphical models, and the rules that guide the modeling process. Defining a complete language is considered a very difficult job. The task eases significantly it is only for one problem domain in one company.

The primary issue for finding domain concepts is the professional provided by a domain expert or a small team of them. The expert is a developer who has already developed many products in this domain. He may have designed and developed the architecture behind the products, or been instrumental for forming the component library. Expert can find the domain concepts from its terminology, existing system descriptions, and component services.

II. RELATED WORKS

The studies of DSML, Frame work architecture code generation using DSML are related topic to this work. DCOM provides starting point to the DSM and Shelf concept enhanced the idea. Meta Case Company develops the tool called MetaEdit+ which provides the idea on Modeling and code generation. The Domain specific modeling3 idea is applied in the mobile, Insurance and watch applications. Companies are aware of this new application and try to modernize them by using this new technology.

III. DSML IMPLEMENTATION IN METAEDIT+

DSM concepts are perfectly applied to the domain semantics. DSM tools one can enter domain concept and can define its properties. In the repository all inputs entered can be stored. Required Modeling concepts in the software production can also be added. A DSM language should follow the rules as they exist in the domain. Once defined, the language guarantees that all developers use and follow the same domain rules. These rules are of different kinds and typically relate to connections between concepts, layering models, reusing designs, etc. A visual modeling language requires symbol definitions. The notation should illustrate as well as possible the corresponding domain concepts' natural "visualization". End-users are often the best people to invent these symbols.

A. Data Model

The data model explains how constructs our design information consists of. In MetaEdit+ the structure of the data model is based on the GOPPRR[4] data model. GOPPRR is stands for **Graph-Object-Property-Port-Role-Relationship**. Each one is called a ***meta-type*. Object** is a little component exists on

itself. Examples are a Button[4], a State, and an Action. All instances of objects hold reusability: An existing object can be reused in the other graphs by using existing function. A **relationship** is at connection between groups of objects. Relationships which connect the objects through roles. A relationship in a "Transition" that belongs to a Watch Application or an Inheritance is one example an object participated in a relationship can explain itself by the Role object. Transition relationship in the Watch Example which specify the objects at the both end. Inheritance relationships there are two kinds of roles one is Ancestors and another is Descendants. One segment of role is a **port** is one segment of an object in which a role can connect. For example, an Amplifier different three port one for analog input, one for digital input, and another for analog output. In Role connecting to each of these will have different semantics. A **graph**[4] explains "collection of objects, relationships, roles, and bindings" The figure given bellow shows the steps to be followed to implement Domain Specific Modeling implementation in MataEdit+. A **property** is a qualifying characteristic associated with the other types, such as a name, an identifier or a description. The above description and diagram explains how DSML implemented in the MetaEdit+ Tool.

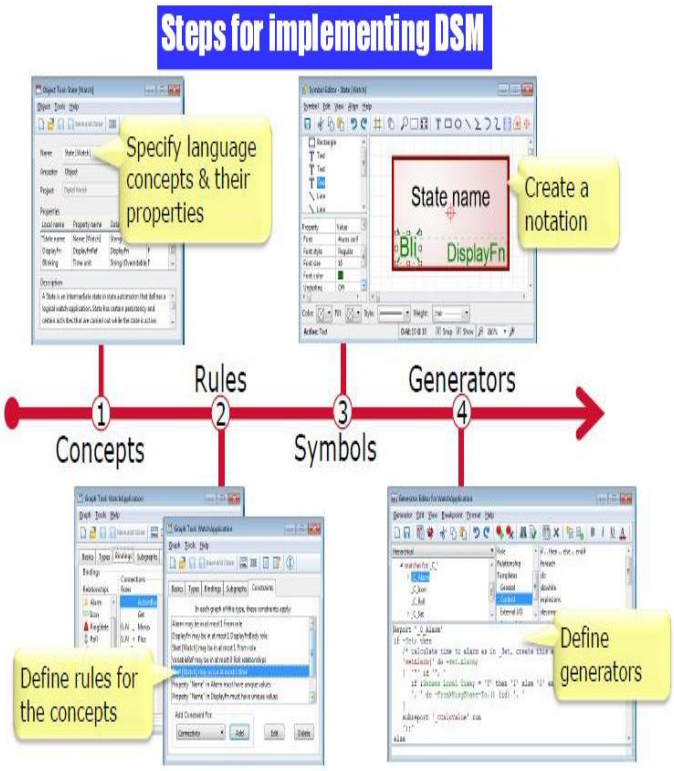


Figure 1 Step to Implementing DSM in MetaEdit+

B. Use of Simulation software

There are numbers of software available which can mimic the process involved in your research work and can produce the possible result. One of such type of software is Matlab. You can readily find Mfiles related to your research work on internet or in some cases these can require few modifications. Once these Mfiles are uploaded in software, you can get the simulated results of your paper and it easies the process of paper writing.

As by adopting the above practices all major constructs of a research paper can be written and together compiled to form a complete research ready for Peer review.

IV. DSML IMPLEMENTATION IN GENERIC MODELING ENVIRONMENT(GME)

In the Generic Modeling Environment (GME) tool[5] the DSM implemented in the browsers which are the very important components. Saved models in the database are arranged in the project order. A project is a group of models created using a particular modeling paradigm[5]. Within the project, the models are arranged into folders. Folders themselves and models in one folder can be arranged in the hierarchical order further standalone models are also be present. Through Model Browser one can view the entire project "at a glance". The Model Browser Controls can show, add and move the Models. Six browsers in the GME are taking base works of GME.

A. Part Browser

The part Browser can be placed any side of the independent window. Based on present working aspect the Part Browser can insert current model. It also views all parts excluding details of connections. Lower of the Part Browser, tabs view the aspects of the present model. By clicking the tab change the aspect of the present model can be selected any one. It also attempts to change the aspect of all the open models. If a particular model does not have the said aspect, its shows that the aspect remains active. The Part Browser can be used to drag a single object at a time and drop it either in any editor window or in the Model Browser**.** If a reference is dragged, a null reference is created because the target object is unspecified. Keep in mind that references (Including null references) can be redirected at any time by dropping a new target on top of them. The Part Browser can be floated as an independent window.

B. Attribute Browser

In Attribute Browser Preferences and Attributes are available in a modeless dialog box. Browser changes are recorded immediately. Changes to toggle buttons, combo boxes[5] and color pickers can be done immediately. Changes in the single line edit boxes are recorded when "Enter" button is pressed on the keyboard In a multiline edit boxes the Enter Key for the new line insertion so hitting the enter key does not updated the value. The object selection for the attribute browser works as follows. The context menu access to Attributes and Preferences, now even from the Model Browser, works. Selecting an object or inserting, dropping or pasting operation will selects that object for the Attribute browser. More than one object is selected – in the Model Browser or in the Model Editor - the attribute browser will allow only the common attributes of these objects. Dialog box there are three tabs, first one for the attributes second one for the preferences and the last one for the properties. The Attribute Browser window, just like the Model Browser window it can float as an independent window, it can be placed in any side of the GME [6].

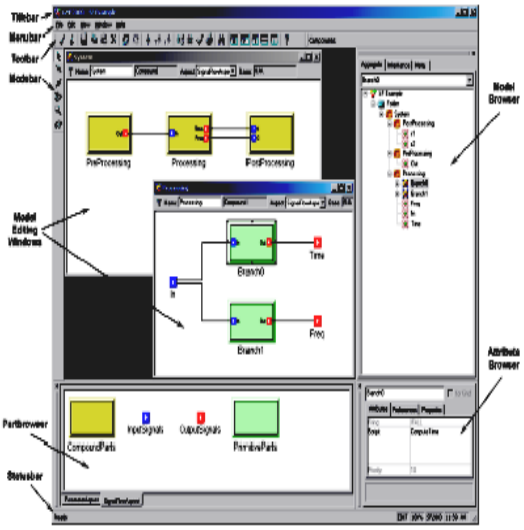
C. Model Browser

The GME[5] is a configurable graphical editing environment. It is configured to work within a particular modeling paradigm via a *paradigm definition file*. Paradigm definition files are XML files that use a particular, GME 6 specific Document Type Definition (DTD). Models cannot be created and edited until a paradigm definition file has been opened. Once a project has been loaded, the GME opens a Model Browser **window**. The Model Browser is primarily used to organize the individual model that leads to overall project. The graphical editor is used for actually constructing the project's individual models.

The vital features of the **Model Browser**[6] are accessible through the three tabs displayed at the top of the **Model Browser**. These tabs deal with the **Aggregate**, **Inheritance**, and **Meta** hierarchies. The **Aggregate** tab having a tree-based containment hierarchy of all folders, models, and parts from the highest level of the project, the Root Folder. The aggregate hierarchy is ignorant to aspects, and is capable of displaying objects of any kind. More information on the aggregate hierarchy will be provided shortly.

Figure 2 Model Browser showing folders and models.

The **Inheritance**[7] tab displays the type of inheritance and hierarchy. Inheritance is entry driven by the current model selection within the aggregate tree. Present method of selection in the aggregate tree in the figure above is a model "Generator Base" is an example. It has one subtype, called "SubGenBase", and two instances, bearing the name "Generator A" and "Generator B". This type of /instance relationship is shown in the Inheritance tab. We can also have an instance model of the "SubGenBase" subtype, called "SubGenBase". The letter "S" denotes a subtype and a letter "I" can be found in front of instances in the **Attribute** tab. The **Meta** tab shows the modeling language at a glance it displays the legally available array of Folders and objects that can be added at any level within the aggregate hierarchy. At the "Root Folder" level we can add "Folder" folders. Within these folders, we can add models primitive and Compound. From these models, more parts can be added. Above discussion explains how DSML implemented in the DME.



GME 5 Main Editing Window

Figure 3 GME Main Window

V. IMPROVEMENT AS PER REVIEWER COMMENTS

The important method incorporated in implementing DSL require reprocessing embedding The Techniques involved in implementing a DSL needs reprocessing, embedding, compiler/interpreter, compiler generator, extensible compiler/interpreter, and commercial off-the-shelf. The language designer has to select the appropriate implementation approach, according to the project to be implemented. The development is influenced by the fact that DSML has to be included in the already existing data acquisition software DEWESoft [8] and this product is developed in Object Pascal, more specifically in Delphi. These limitations lead to decide for compiler/interpreter implementation approach, where some of the compiler generator tools were used. Commonly the idea of a lexical analyzer is simple implement. But the construction and implementation of a lexical analyzer is time-consuming. Hence, in the construction of a lexical analyzer, a compiler generator implementation approach can be used to speed up this process. In the case of the Sequencer, the syntax and semantic analyzer has been developed independently of existing compiler generator tools. The syntax of the Sequencer was described using standard BNF notation.

Starting non-terminal NT_START, it can be seen that the reserved words like Begin, End embody DSL statements that represent functionalities to be performed from the measurement. Another advantage can be observed if the Sequencer programs are compared with the DCOM[8] application with the number of lines of code. The size of code (LOC) is presented for five different applications developed with Sequencer and DCOM objects. All Sequencer programs and DCOM applications have

the same functionality. The advantage of Sequencer compared to the API solution, since the Sequencer programs were at least 13 times shorter than the same DCOM applications.

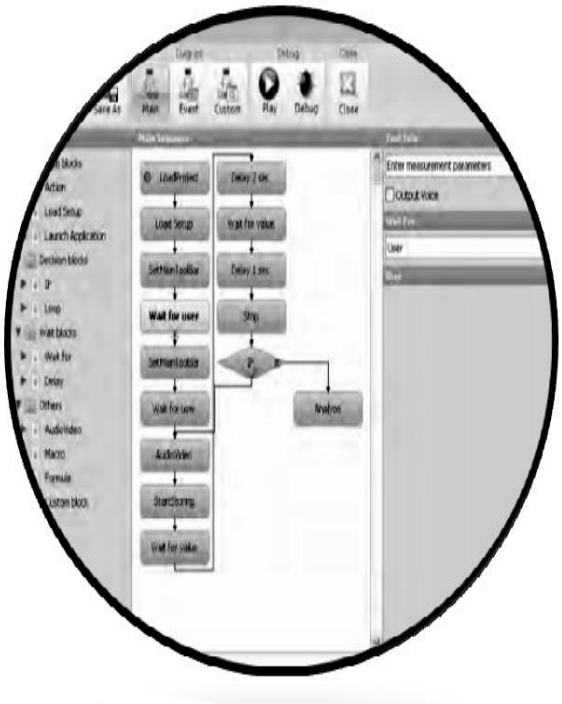


Figure 4 Sequencer's modeling environment

VI. CONCLUSION

GME provides balance between metamodel[9] precision and validation. Analyzing the tools and DSML implementation gives ideas to identify necessary steps to implement the DSML in a Tool... Graphics, attributes, rules, relations, browser and generator are identified common features of DSML to be implement in a tool. Some other specific characteristics are present in the Tool to enable domain specific nature. DSML provided number of advantages which reduces the cost and speed of the process of Development paradigm.

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in American English is without an "e" after the "g." Use the singular heading even if you have many acknowledgments.

REFERENCES

[1] "*Domain-Specific Modeling and Model Driven Architecture*", January 2004, Steve Cook, Software Architect Enterprise Frameworks & Tools Group Microsoft Corporation.

[2] "*Introduction to the 3rd workshop on Domain-Specific Modeling*", Juha-Pekka Tolvanen Jeff Gray.W.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

[3] "*Handling Crosscutting Constraints in Domain – Specific Modeling Communications*" Jeffry Gray, Ted Bapty, Sandeep Neema, James, James Tuck of the ACM, October 2001.

[4] www.metacase.com

[5] GME 2000 User's Manual, Institute for Software Integrated Systems, Vanderbilt University.

[6] www.vanderbilt.com

[7] *Improving Domain-Specific Language Reuse with Software Product Line Techniques*", Jules White, James H. Hill, Sumant Tambe, Aniruddha S. Gokhale, Douglas, Douglas C. Schmidt and Jeff Gray July/August 2009 IEEE Software.

[8] "*From DCOM Interfaces to Domain-Specific Modeling Language: A Case Study on the Sequencer*" TomaZ Kos, TomaZ kosar, Jure Kenz, and Marjan MernikComSIS VOl.8, No. 2, Special Issue, May 2011 378

[9] "*Domain-Specific Modeling: Enabling Full Code Generation*" by Stevan Kelly and Juha-Pekka Tolvanen, March 2008, Wiley-IEEE Computer society Press.

[10] Juha-Pekka Tolvanen Matti Rossi MetaEdit+: defining and using domain-specific modeling languages and code generators OOPSLA '03 Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications
Pages 92-93

AUTHORS

Ramesh Ramanathan is pursuing his Phd in Department of Computer Science and Engineering from Pondicherry University. He has completed his M.Tech (Computer Science and Engineering) from Pondicherry University. His research interest includes Domain Specific Modeling Language.

Dr.V.PrasannaVenkatesan is working as a Associate Professor in the Department of Banking Technology from Pondicherry University. His areas of specialization are Object Oriented Modeling and Design, Banking Technology, Service Oriented Architecture, Smart Banking.