IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 3, March 2015.
www.ijiset.com
ISSN 2348 – 7968

Time scheduling system using domain specific modeling

Chanchal Jayaraj[1], Ramesh. R[2], Prasanna Venkatesan. V[3], Martin. A[4]

[1]PG Scholar, Sri Manakula Vinayagar Engineering College, Puducherry, India.
[2] Sr. Programmer (Finance and Accounts Section) Pondicherry University, India.
[3]Associate Professor, Department of banking Technology Pondicherry University, India.
[4]Associate professor, Sri Manakula Vinayagar Engineering College, Puducherry, India.

Abstract- Domain-Specific Modelling (DSM) is a software engineering methodology for designing and developing systems such as computer software. It involves methodological use of a graphical domain specific language (DSL). DSM language raises the level of concept beyond coding, building development process faster and easier so it requires less effort and less low-level details to specify the system model. When companies develop their own product in-house DSM solution domain-specific modelling languages and code generators they often need to offer evidence that it gives better results than their current development model. We describe an approach applied at digital watch model to evaluate a DSM solution for developing embedded devices. The evaluation come up to takes into report the objectives set for the creation of the DSM solution. The evaluation proved the benefits of the DSM solution, an increase of at least 70% in developer productivity, and significantly improved value of the code and model development process. The final products are generated from these high-level conditions. In this paper we describe why DSM is faster and how to build a DSM language and generator using MetaEdit+ tool.

Keywords— Domain Specific Modeling, Time Optimization, Productivity and Output.

I. INTRODUCTION

In recent years, software development meets many problems that are caused by increase in scale the complexity of software design, the reduction of development schedule and cost and so on. Domain specific modeling (DSM) as a method to solve these problems has been attracting methods. Domain-specific modeling (DSM) is a higher level of CASE process, a way to model data structures and logic in domain concepts independent from programming languages and thus also include syntax details. The final source code in a desired programming language is derived automatically from these high concept models by using exact language generators. This is possible, because both the modelling concepts and language generators are defined by users of DSM solution software. The basis for DSM is Language Engineering allowing to define and to use various Domain Specific Languages to implement. The whole process of Meta-modeling in the MetaEdit+ tool rotates around the Meta types represented together as GOPPRR. It stands of Graph, Object, Property, Port, Relationship and Role. Meta modeling starts with defining objects and its properties. After objects are generated a relevant diagram is assigned to the object. The created, assigned diagram can be drawn in the editor or imported from outside. By default the connection point on the diagram is its centre but this can be changed by handing over different ports on the border of it. Ports are point where relationships are attached to the object. GME supports various concepts for edifice large scale, complex models. They comprise hierarchy, multiple aspects, sets, references, and clear constraints.

II. DSM DOMAIN-LANGUAGE

Software development was mainly shifted from the Assembly language to high-level third generation language (3GL) like BASIC. This led to about 500% of productivity on average also migration from BASIC to Java has also caused an improvement of about 25% in development productivity. The modeling language is based on domain concepts identified during the domain analysis process. For example, Digital watch domain might disclose such concepts as labelled, buttons, icons, and time unit for the predefined tags and symbols are used in creation of model. The modeling language thus probably contains such concepts, beside with ways to association them together. The objective here is to map the chosen concepts correctly to the domain semantics. Domain-specific language differs from earlier code generation attempts in the tools of the 1980s or UML tools of the 1990s. Both code generators and modeling languages were built by tool vendors like metaCASE designer or any other modeler. While it is possible for a tool vendor to create a domain-specific language and generators, it is further normal for domain-specific language to occur within one group. One or a few expert developers creates the modeling language and code generators, and the rest of the developers use the created modeler language and develop the products as they need.

PROBLEM DOMAIN	SOLUTION DOMAIN/GENERATION TARGET
Telecom services	Configuration script
Insurance products	J2EE
Business processes	Rule engine language
Industrial installation	XML
Medical device configuration	XML
Machine control	3 GL
Call processing	CPL
Geographic information system	3GL,propriety rule language, data structure
SIM card profiles	Configuration script and

ISSN 2348 – 7968

	parameters
Phone switch services	CPL, voice XML,3 GL
E-Commerce marketplaces	J2EE,XML
Automation network	C
Crane operations	C/C++
SIM card applications	3GL
Applications used in Microcontroller	8-bit assembler
Household appliances features	3GL
Smartphone UI applications	Scripting language
ERP Configuration	3GL
Handled device application	3GL
Phone UI applications	C
Phone UI applications	C++

Table.1 DSM-Domains

III. DSM TOOL METAEDIT+

Defining the modeling language deals with identifying the modeling idea, the rules and idea that make the use of language and apply accuracy of models is used to develop, and the notation used to present these in models. These are usually best found from the domain terminology, system architecture, existing system descriptions, and component services. For the language implementation, MetaEdit+ provides a Metamodeling tool suite for entering the modeling concepts, their properties, associated rules and symbols. This definition is stored as a Metamodel in the MetaEdit+ repository allowing future modifications, which reflect automatically to all related models and generators for code. The Metamodel elements define parts of the watch specific modeling language. MetaEdit+, which is a fully functional CASE environment with wide modelling language support, documentation reports and code generators. The help manual provides information about MetaEdit+ tool. It is possible to extend the external interfacing, import and export features of MetaEdit+ with the API add-on product. MetaEdit+ forms a multipurpose and powerful multi-tool environment that enables flexible creation for product model, maintenance, manipulation, retrieval and representation of design information among multiple developers.

IV. CODE-DRIVEN AND MODEL-DRIVEN DEVELOPMENT

Developer generally differentiate between modeling concepts and coding of the product. Models architecture are used for designing systems, understanding them better specifying requirement functionality and creating documents for upcoming use. Code is earlier written to implement the designs. Debugging, testing, and maintenance are done on the code level too. Quit often these two different approaches are irrelevantly seem as being rather divided, although there are also various ways to support code and models in different ways. The single most important asset of a DSM environment is the domain-specific modeling language the result of the increased productivity of the developers working with the DSM environment. While the code generator and the domain frameworks are important parts of the DSM environment.

V. DOMAIN SPECIFIC CODE GENERATOR

Generator translates the models into the required output. The code and model combined which produce generator (code + model=generator). Domain Specific Modeling is based on domain specificity and automation it's specify the solution in a language that use directly rules and concepts, the second is complete automated generation of the final product. The patterns for generators by type of output are.

- Simple text
- Model checking
- Documentation
- Xml
- Flow machine
- State machine

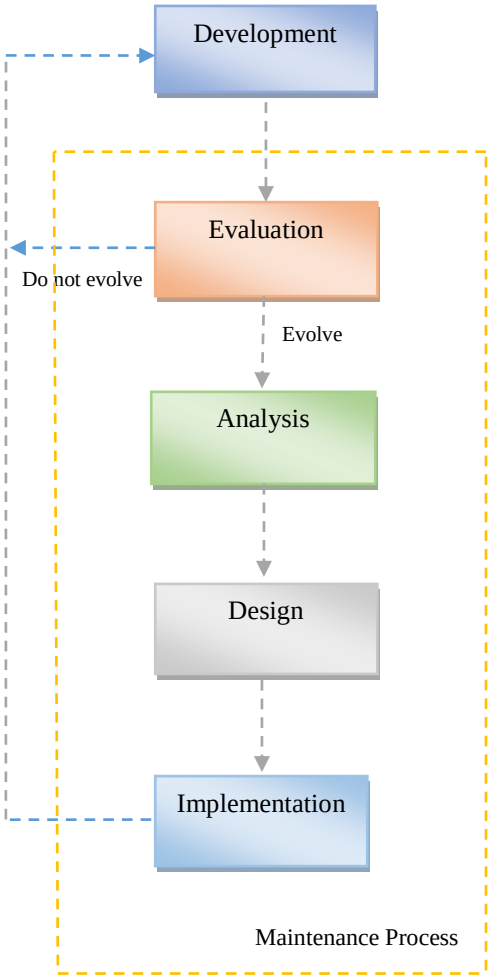

Fig.1 The process of maintenance & code generator

ISSN 2348 – 7968

The key issue in building a code generator is how the models concepts map to code. The domain framework makes this task easier by raising the level of abstraction on the code side. In the simplest case, each MetaCASE modeling tool contains symbol that produces predefined certain fixed code it can be run manually or in the automatically, including the values entered into the symbol as argument. The generator can also generate different code depending on the values in the symbol, the relationships it has with other information in the model. The different level of generator are:

- Generate per file to be generated.
- Generate per section in a file.
- Generate per Metamodel element

VI. BUILDING SUPPORT FOR DOMAIN-SPECIFIC MODELING

A software development environment to collect DSML application data we utilize MetaEdit+ to collect DSML application data. MetaEdit+ tool is a client server based development environment to develop software by DSML it can measure the quality of DSMLs and analysis possible options for DSML evolution. DSML developers implement the DSML designed in the design phase and enable domain experts to use the DSMLs.

Building support for DSM with full automatic code generation include phases are:

- **Assembling the Domain-Specific component library:**
  It facilitates code generation development meaningfully code component can be either developed in-house or as third-party components.
- **Developing the Domain-Specific Modeling and Domain-Specific language and Editor:**
  The domain experts combine knowledge and experiences from multiple resources from component library, rules and architecture
- **Developing the Code Generator:**
  The Domain experts specifies how code using the component can be automatically generated from the structure in the user's model. The DSM model tool should provide the necessary functionality for creating such generation of code script, and should guide to use the model. The end user should be able to use the code generation simply.
- **A Language Developer:**
  It define or enhance a DSL in accordance with the needs of product developers. The language developer not only defines the syntax of modeling language also describes its meaning in terms of semantics and ensures that newly added concepts are properly integrated in the exiting language and provides a manual for their user.
- **Tool Developer:**
  It develops code generator for the DSL which includes the generation of production and test code as well as the analysis of the content and its quality of the model architecture and generated code. It integrate newly developed or reused language processing components and generator to form tools used in project.
- **A library Developer:**
  It develops software component or library to simplify the code generator. It is related to the tool developer but requires more domain knowledge.
- **The product Developer:**
  It employ the provided tools for different activities within the project

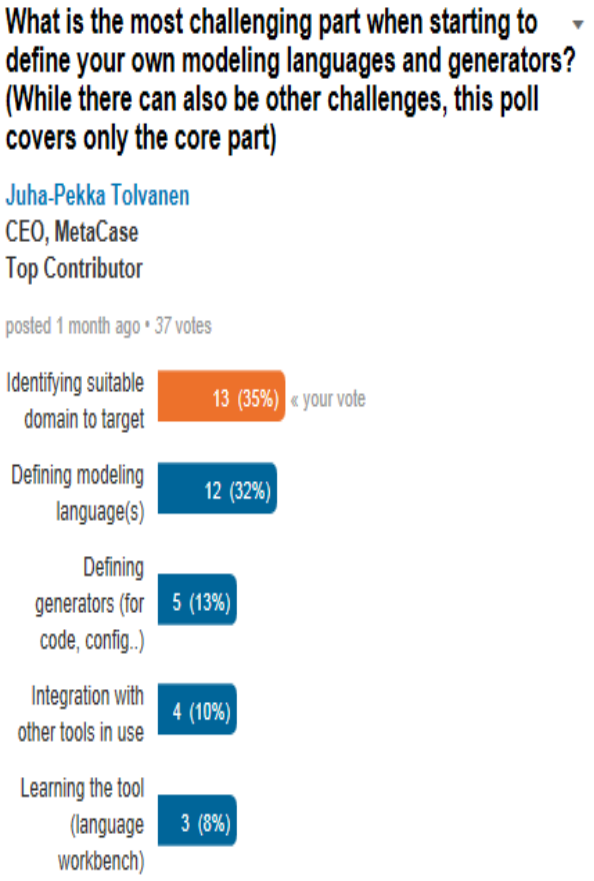

Fig.2 Graphical analysis of reports

VII. PRODUCT STUDY

Basically the Data model forms a higher level structure defining the structure of both the design information and the method information stored in the repository. With this single data model, design information is shared between the various tools and editors and methods are integrated. The basic features of the multi-user version of MetaEdit+ are modeling language deals with identifying the modeling concepts, the rules that constrain the use of language and enforce correctness of models, and the notation used to present these in models. These are usually best found from the domain terminology, system architecture, existing system model description, and component service of the tool. MetaEdit+ provides a metamodeling tool suite for entering the modeling

ISSN 2348 – 7968

concepts, their properties, associated rules and symbols. Alternatively you may specify the metamodel using graphical metamodel in the MetaEdit+. The language definition is stored as a metamodel in the MetaEdit+ repository allowing future definition is stored as a metamodel in the MetaEdit+ repository allowing future modification, which reflect automatically to models and generators. In existing system consist of lesser productivity than domain specific modeling tools, because of lacking in the stage of designing, coding, implement

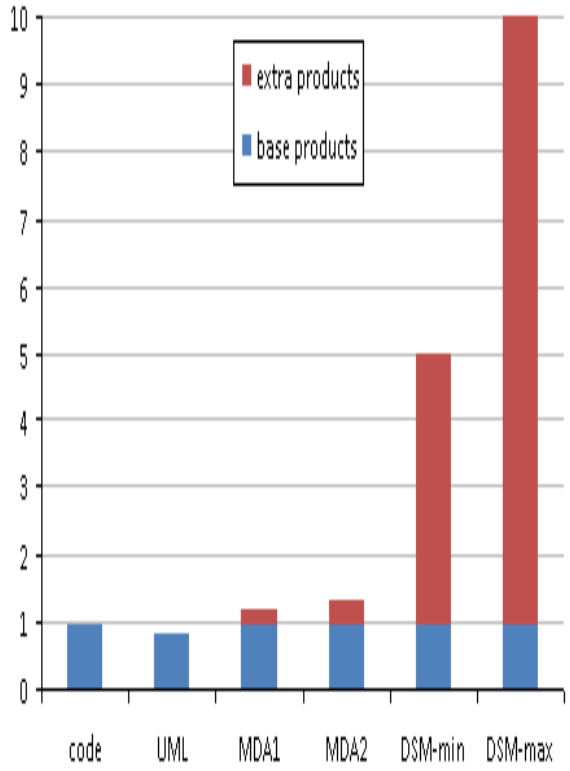

Fig.3 graphical reports of analysis

VIII. CONCLUSION

We proposed a quantitative approach to the evolution of DSML by application data. This approach requires DSM and DSM tool, Domain-Specific Modelling allows quicker development for the DSM solutions, which based on models of the product, rules and relationship rather than on models of the code generation. Work model experiences of DSM show major improvements in output, lower cost of development and better quality for the designed product MetaEdit+ is providing Meta modeling technique. GME is also a fantastic tool that supports DSM principles. It helps the system designers in recognizing system models. It provides tool support for developing Meta Model. MetaEdit+ reduces the time and cost for the developing model, also needed down to the order of days or weeks, And DSM also provides a better role for expert developers. The product study gives many ideas about the tools in view of the modeling technique and tools used, security of the product development, class and inheritance, methods, attributes and user friendliness of a tool. The output of the tool which represent UML class diagram aspect diagram and attribute diagram are providing very good understanding of the Meta model. GME code generator generating code in C++. Object Oriented Technique is followed in all DSM tools and code generator it generate the code and design model for object oriented languages. The following important concepts may incorporated in the future enhancement of the DSM Tools are Reverse Engineering Technique, Rollback mechanism, Documentation, Security, Improve the User friendliness.

REFERENCES

[1] Kelly,s. Tolvanen,j.p, domain-specific modeling: Enabling full code generation, wiley, 2008.
[2] Kellys, s., pohjonen,r.worst practice for domain-specific modeling,ieee software ,jul/aug,2011.
[3] Safa, the making of user-interface designer: a proprietary dsm tool, oopala workshop on domain-specific modeling, 2010.
[4] Pohjonen, r., kelly, s., “domain-specific modeling,”dr. Dobbs journal, august 2011.
[5] http://dsmbook.com.
[6] Metaedit+ reference manual
[7] Metacase. Domain-specific modelling with metaedit+. url:http://www.metacase.com/.
[8] Microsoft visual studio 2005 sdk including domain Specific language tools main page, http://msdn.microsoft.com/vstudio/dsltools/ 2007.
[9] Metacase, metaedit+ version 4.5, the graphical metamodelling example.
[10] Nokia,www.metacase.com/papers/metaeditnokia.pdf
[11] Metacase, metaedit+ workbench 4.5,5.0 sr1 user, guide, http://www.metacase.com/support/45/manuals/2009.
[12] Domain-specific modeling with metaedit+: 10 Times faster than uml - white paper.
[13] Polar, proceedings of domain-specific modeling,08 www.dsmforum.org/events/dsm09/papers/karna.pdf.
[14] MetaEdit+: Defining and Using Integrated Domain Specific Modeling Languages
[15] Panasonic, proceedings of domain-specific modeling, www.dsmforum.org/events/dsm07/papers/safa.pdf.
[16] Juha-Pekka Tolvanen, Jeff Gray, Introduction *To The 3rd Workshop On Domain-Specific Modeling*
[17] Jeff Gray, Ted Bapty, Sandeep Neema, James Tuck, *Handling Crosscutting Constraints in Domain-Specific Modeling* Communications of the ACM, October 2001.
[18] Jonathan Sprinklea, Gabor Karsaib A *domain-Specific visual language for domain model Evolution*, Journal of Visual Languages and Computing.
[19] J.C. Grundy, J.G. Hosking*, R.W. Amor, W.B. Mugridge, Y. Li *Domain-specific visual languages for specifying and generating data mapping systems,*

ISSN 2348 – 7968

Journal of Visual Languages and Computing 15 (2004) 243–263.

[20] Sagar Sen "Domain Specific Modelling and Simulation of Physical Systems" MSc. (Thesis) Student School of Computer science McGill Uni,.

[21] A. van Deursen and P.Klint, "Littke lnguages": little maintiance?", In Journal of Software Maintenances, Pp.75- 92, 1988

[22] Thomas J.McCable,"A complexity Measures", In the Proceedings of the 2th International Conference on Software Engineering, pp 308-320, 1976.

[23] D. C. Schmidt "Guest Editor's Introduction: Model Driven Engineering," IEEE computer, vol. 39, no. 2, pp. 25-31, 2006

[24] Defining Domain-Specific Modeling Languages to Automate Product Derivation:Collected Experiences Juha-Pekka Tolvanen and Steven Kelly

[25] Lecture Notes in Computer Science Springer Berlin / Heidelberg Kalle Korhonen Volume 2510/2002.

[26] Uwe Zdun *Reengineering to the Web: A Reference Architecture*, Proceedings of the Sixth European Conference on Software Maintenance and Reengineering (CSMR.02) 1534-5351/02 © 2002 IEEE

[27] Emanuel s. Grand Krish Narayanan *Rigorously Defined domain modeling language* Department of Computer Science, University of North Dakota.

[28] Jan Heering Marjan Mernik Anthony M. Sloane, *Domain-Specific Languages.* Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03) © 2002 IEEE

[29] Ph.D thesis of Jeffrey G. Gray "*Aspect-Oriented Domain-Specific Modeling: A Generative Approach Using A Metaweaver Framework" Editor, DSM fifth conference*

[30] http://en.wikipedia.org/