Survey on model driven development

Sina Tebiani
Mälardalen IDT department
Intelligent embedded systems
Sti10003@student.mdh.se

ABSTRACT

Model driven development has a large impact on the development process cycle of software these years. Although it is not replaced totally with the traditional implementation methods but it is going to force itself to the industry by providing benefits for software production process. This new technology is used to create the models to generate the implementation of whole system that is under construction. In last fifty years the researchers and developers have done many efforts to create abstractions that help to work in design environment rather than computational environment. Although the third generation of languages such as java and C++ aimed to elevate the abstraction level of software developing but still there were inabilities to reduce the platform complexity and to express the domain concepts effectively. To address these inabilities the model driven engineering technologies were developed. Many companies try to get the benefits of MDD (Model Driven Development) more than last decades. This paper will survey on the model driven engineering and declare the state of art of this technology and the facing challenges and introduce different tools in this area.

Keywords

Model Driven Development (MDD), Model Driven Architecture (MDA), Model Driven Engineering (MDE)

1. INTRODUCTION

Model driven development (MDD) is named model driven software engineering (MDSE) or model driven engineering (MDE) usually is the methods to create abstractive models as the first level artifacts and transformation of these models to generate source code [11]. Many early languages like assembly and FORTRAN raised the level of abstraction in the early years but even they were more in computing environment instead of design environment that was making platform complex and less observational. Model driven engineering has been one of the common applicable methods in software development process in recent years. The complicated systems include many different complex functionalities and lines of codes, The model driven engineering helps developer to solve the problems and complexity of system with limited efforts and small budgets. In the IT environment, with fast speed technology development, the requirements for producing new complex systems are changing. To cope with this fast changeability and complexity the model driven engineering provide a new development methodologies that reduce the cost and time severely much effective than before. The model driven development has changed the software production industry form the code centric base to the model centric base by raising the abstraction level of the domains. [2] using the graphs to model the domain concepts not only solve unambiguity and decrease faults even it helps an wider range of experts specialties such as system engineers and software architectures and testers to raise the reliability, maintainability of the product that meet the user’s needs. In this paper we will introduce the state of art of the MDE in recent years and in part 3 the fundamental elements of model driven software development will be presented and part four covers the languages for model driven development discussion. Part six will discuss about requirements of MDD and Part 7 consists of conclusion of all discussions and the references will be highlighted in part 8.

2. STATE OF ART OF MODEL BASED ENGINEERING

Regarding the model driven development advantages impact in industry, many companies changed their development process methodology to model driven development. The IBM is one of the big leaders in this field deployed the MDD as new and key part of their development strategies. Object Management Group (OMG) is one of the organizations that with the big interest in MDD changed the strategies of system development with direct address to better understanding, design, architecture, deployment and modification with providing model based development frameworks. [2] The approach of OMG to MDD declared with some specific standards that gives the ability of transformation into other models and defining model specifications. [2] The OMG recently uses the Unified Modeling Language(UML), Object Constraint Language(OCL), Meta Object Facility(MOF), XML Metadata interchange(XMI), Common Warehouse Metamodel(CWM) standards. [2] Besides the OMG’s definitions the EMF (Eclips Modeling Framework) and JMI (Java Metadata Interface) propose some other standard definitions. [2]

2.1 MDD TOOLS

The model driven development intends to increase the abstraction level of the development process. The OMG (Object Management Group) is a leader in providing such efforts to define new
standards. The leader companies and other open source projects develop new technologies to approach the concept of MDD. [4] With raising demands for MDD, new tools are produced to response to interest of the market. Some companies focus on customer and development process coverage such as Borland (Together), Softeam (Objenteering), and IBM (RSA) [3]. We can mention other products with less coverage of development process like Xactium(XMF-Mosaic) and SparxSystems (Enterprise Architect). None of these tools are complete exactly in respect of features and characteristics although some of them provide acceptable support for modeling and UML.

3. BASIC ELEMENTS OF MDD


3.1 Modeling Language

As we know modeling languages will increase the abstraction level of tools to better understanding of system model. We can specify the modeling language to two parts: GPL (General purpose languages) and DSL (Domain Specific Languages). For instance the UML is the general purpose language that is used from the early phases of software development until to the implementation. The DSL languages have been introduced to specific domains to reduce the ambiguity and increase the efficiency of modeling in all life cycle of software development.

3.2 Domain Knowledge

Domain knowledge plays significant role in software development. The software development is exactly the process that starts with domain aspects and designing the system fundamentals based on these aspects to end with implementation. Well understanding domain aspects will help the developers and designers to create the applications and structures more precise. The deploying new techniques, new methods and new algorithms need the good knowledge of domain concepts even in solving errors during the development. The expertise in the specific domain can increase the preciseness of the model and create the well-defined DSLs. [3] knowledge of application domain plays major role in development process.

3.3 Meta-models

The meta-model is basically construction of gathered concepts of specific domain. [9] The model defines an event in the real world while the meta-model is the other abstract level of model which mentions to the properties of model. [9] It is used commonly in the model driven software design. The Meta Object Facility (MOF) is a meta-model standard that has been defined by OMG to create models. MOF has four layers architecture. The meta-model layer is the highest layer that is called M3 [10]. The M3 as a language is used by MOF to define meta-models in the layer M2. The UML is layer2 meta-model. The M2 layer defines the elements of the M1 layer. The last layer M0 is the description of real world objects. [10] The figure below shows the infrastructure of the MOF technology.

![OMG Modeling](image)

3.4 Model Transformation

The input to the transformation is model and output can be model or different level of executable codes. [5] We can separate three different categories for model transformation [3]. Refactoring transformation, model to model transformation and model to code transformation are three major parts of MDD transformation process.

Refactoring transformation: is the revision version of original model for example the renaming the instances of UML entities or substitution of class with a set of classes and relations in the diagrams and meta-models that define them. [5]

Model to model transformation: is the model transformation form one type of model to another type of model or set of models. For instance we can mention to the conversion form the set of entity classes to Plain Old Java Objects (POJOs), XML formatted mapping descriptor files.

Model to code transformation: it is the conversion of the model to the code fragments. [5] For instance the C++ and JAVA uses this conversion to get coded outputs. But these outputs are not limited to the programming languages. Configuration, data definition and other type of data can be generated form the models that is described in UML. [5]

3.5 Standards

One of the elements in MDD is the standard. OMG proposes the model driven architecture standard of notations and tools for model driven development. These standards basically are the ability of MDD environment to model specifications and transformations to other models.

3.6 Formal Methods
Formal methods are used to verify development process by providing mathematical techniques to describe analysis and design models. Formal methods elevate the level of correctness and completion of MDD. For example the Π-Method is one of these formal methods that are used in software development process.

Π-Method is the new formal method that has been designed in ArchWare European Project to be applied in MDD. [3] We can mention other formal methods such as Z, VDM, FOCUS and B but there are some differences with Π-Method. These methods support model driven development process but they do not support architectural development while Π-Method can support dynamic architecture. [3]

The mathematical formulation that describes and analyzes the computations is based on π-calculus.

3.7 Tools

Many kinds of tools have been designed in model driven development area to support reliable abstraction and automation. Some of them are open source projects like Model Driven Development integration (MDDi) project Generative Modeling Tools (GMT) which supports different tools e.g. Open Architecture Ware, Atlas Model Weaver (AMW) and etc. [3]

In addition to these open source projects we can mention to commercial tools like Together, OptimalJ and Objecteering.

Together is a product of Borland with all sets of MDA abilities and including set of tools to organize, run and debug the transformations. [3]

Objecteering is designed with Softeam. This tool set by using the language “J” which is simplified level of Java provides wide extension mechanisms that specify UML for specific domain or platforms. [3]

4. Languages for MDD

In the model driven development the models play significant roles in development. The idea of MDD comes from the need of automatic codes generation from models. This idea has some benefits: 1. the generated codes are the precise comprehension of models and their properties. 2. The modeling increases the speed of coding and eliminates the tiredness of coding manually. [8] To approach to these advantages the semantics and syntaxes of modeling languages needs to be as precise enough as traditional programming languages. Recently the UML is the modeling language that had major impact in the MDD improvement. In some cases modeling languages are not able to specify some details to the implemented codes from models. In these cases the heterogeneous models technique is used. [8] In this technique while the high level modeling language abstract the concepts, the specific details are specified by detail level language that consists of fragments that are embedded into the part of models. [8] For example the state machine is the modeling detail language with textual syntax that computational details are specified in the transitions from one state to other one. This combination of high level model languages and detail level language is called action languages. [8] These action modeling languages provide the possibility of using MDD during the full development cycle.

5. Challenges in MDD

MDD is completely different from the traditional software development process. The MDD normally focuses on the model usage and abstractions to create automatically final source code while the traditional methods relies on codes and documents but MDD still is not replaced hundred percent with that ways. It means that still there are a lot of challenges and problems with establishing the MDD in companies and organizations. In this part we discuss four main challenges within adopting MDD technology.

5.1 Life Cycle Issues

Model driven development needs a management of redundant artifacts for abstract and concrete syntaxes. Versions’ of models should be stored and controlled by users. The interrelation between these artifacts is confusing but the solution can be like the storing one syntax and compute all others. [6]

5.2 Model Comparison

All development projects store the versions of artifacts. This versioning needs strongly the comparison between models. The models include graphs, forms, property sheets and texts. The comparing tests are understandable in other cases it is not easy. [6] Modeling tools has done meaningful jobs in this case such as MagicDraw that shows the differences in hierarchical trees and identify the modified parts in diagrams but other tools that were generated in Eclips Modeling Framework have not any solutions for comparison yet. [6]

5.3 Model Transformation

As we know the output of the model transformation is model or code and almost all MDD tools support these transformations but the only difference is the amount of these supports. The challenges with transformation can be mentioned in three categories. [6]

1. Inabilities in result refining:

The models will be broken if the results such as adding codes in protected areas are changed because the refined results are not under version control and generated by managed built environment. [6]

2. Inabilities in result referencing:

We cannot reference the result from other models. In modeling all the entities should be stable otherwise the retransformation will break all references to elements. [6]

3. Inabilities in small changes transforming:

It is not possible to transform small part of the models after small changes to models. [6]
Some tool frameworks propose solutions to these problems, for instance OMG presents refinement attaching to the input models. It is shown in the figure below the capability of the different tools to deal with these problems.

![Tool and framework capabilities relative to model transformations](image-url)

Figure 2 [6]

### 5.4 Model Level Debugging

When the models are transformed to runtime artifacts, the debugging problems raise. Only few tools support model level debugging. The traceability record in model transformation is able to keep track of which model elements mapped to which model elements due to run time but still there is issue with code generator of frameworks that cannot deal with model debugging. [6] For example UML that has the code generation abilities but it does not provide support for debugging for class diagram and sequence diagram that address the source code in Java or C#.[6]

### 6. Requirements for MDD

The companies aim to boost productivity. It is one the motivations for using MDD. The MDD makes this goal in two aspects: [7]

1. Increasing short term productivity of developers: by elevating the amount of the functionalities that primitive software artifacts deliver. The MDD tool vendors give more effort on the automation the production of the code from the models. These efforts can increase the short productivity of developers by delivering more executable functionalities. [7]

2. Increasing the long term productivity of developers: by reducing the obsolescence rate of primitive software artifacts. To approach this aspect of MDD the tool vendors try to increase the longevity of primary software artifacts. More longevity can be cached by increasing the time of value of software and more values need lowest sensitivity of artifacts to changes. [7]

We can consider four major changes than can affect the short term and long term productivity of softwares: [7] 1.Personel 2. Requirements 3. Development platforms 4. Deployment platforms

### 6.1 Personal

In all development processes the particular developers have particular roles and specific tasks that depend on their knowledge. By substituting these people in fact their knowledge will be moved out. To preserve these lost knowledge and to be sure that they will be available and understandable to other developers and designers also the concepts of designed system need to be precise and well-defined for stakeholder specifically to customers. [7] To achieve these goals visual modeling language is the next solution.

### 6.2 Requirements

The requirements for the project are not constant factor in software developing. These requirements will be changed along the development process with considering the more features and characteristics that should be extended in the system. Although these extended features improve the functionality of the products but the impact of these changes to the systems should be limited. For example in the online systems the offline extension is not possible. The dynamic addition of new type at runtime is used to get this approach. [7] To catch this aim we need dynamic extensible modeling languages.

### 6.3 Development Platforms

The initial software artifacts are created in their proprietary tools that the life time of these artifacts depends on the life time of tools. Recently the development platforms are evolved that raising the problems along the platform dependency to their artifacts that has been developed by these tools. To deal with this problem the primitive artifacts should be decoupled from their first platform to transform to other platforms. [7] In addition, the format of the artifacts should be supported by different tools that demand the high level of interoperability. [7]

### 6.4 Deployment Platforms

The developing tools are changing very fast. Developers should be adapted to new platform as fast as possible while they were trying to be mastered in the last one. The software artifacts needs to be free of these changes and it means that the process of obtaining specific platform software artifacts from independent platforms should be automated by user definable mapping. [7]

### 7. Conclusion

Recently the MDD has been improved and has changed the software development cycle from the code centric development to the model centric. This paper tried to have brief survey on MDD and its impact in industry and academia. The state of art of the MDD has been reviewed and discussed. The most important elements of MDD that construct the foundation of MDD were discussed. The level of languages for MDD was expressed briefly just to give an overview to understand how modeling languages work in the MDD. The challenges in this area are discussed and it offers solutions and future efforts that the research and academia needs to go through it for involving in MDD benefits as much as
possible. Many efforts are needed to be accomplished to raise the impact of model driven language more than previous years. To approach this level of inclusion of MDD it needs some requirements that they were presented in this paper. In general the MDD has not founded its role exactly and it needs more training of experts. Tool selecting and adopting these tools to different environment are still challenging but industry expects in the few years the model driven development will be replaced with traditional development techniques and the tools will be improved properly.

8. REFERENCES

[4] Liming Zhu, Yan Liu, Ian Gortonand and Ihor Kuz, Tools for Model Driven Development
[8] Bran Selic, Model-Driven Development: Its Essence and Opportunities,
[9] Tim Cheng, Editor in Chief IEEE, Design & Test Metamodelling for model based system design,
[11] Susanna Teppola, Päivi Parviainen, Juha Takalo, Technical Research Centre of Finland, Challenges in the Deployment of Model Driven Development,