From Model to Code: A Review of Model-Driven Paradigms

Azhar Ali Khaskheli[1]*, Saqib Ali Khaskheli[2], Anum Amir Khwaja[3], Muhammad Yaqoob Koondhar[4]

Abstract: **Several paradigms come under the tree of Model-Driven Architecture among which Model-Driven Engineering (MDE) is a software engineering approach that enables us to concern about the model rather than the writing program. Though models are considered to be the artifacts of the MDE, and designing model at the level of abstractions reduces the headache of the developer of platform specificity. With the help of this behavior of MDE complete view and functionality of the system can be understood before actually building it. When the particular model is designed, that model can be transformed into another model or specific platform source code. Hence this paradigm reduces the time of developer of writing large codes and also saves us from bugs that could occur during the implementation phase when developers writing the code the program, because a human can make but computers can't unless they are directed to. In this paper, several papers have been reviewed relating Model-Driven paradigms, that elaborates model-driven paradigms more specifically model-driven engineering, and different approaches that come under the umbrella of Model-Driven engineering and how they work. The role of code generators has also been described and their practical demonstration is presented that how a model of UML sequence diagram is transformed into source code.**

Keywords: **Model-Driven Architecture, Model-Driven Engineering (MDE), source code, Model-Driven paradigms**

INTRODUCTION

With Passage of time, As Software Systems are evolving, they are becoming more and more complex, and Software developers are facing number of challenges in the software development. Reports says that 60% of the software systems are failed because of lack in implementation which causes bugs in software systems[1]. There can be number of reasons for this lack in implementation however one that is very common is mistakes of the developers and hurry in finding generic solutions. Although after a lot of hard work of software developer, the system which he develops, works only on specific platform though if we want to run it on different platforms we need to develop it for that system which need expertise of software developers in that platform as well. However this platform dependency causes companies a lot of cost in term of money or time[2]. MDA was released and standardized by Object Management Group (OMG), many other model driven paradigms fall under the umbrella of MDA among all models are considered to be artefacts. MDE is one of the approach of software engineering, which suggests us to design models of the particular system, rather than program in particular language, with an abstraction level[3]. This approach of designing the model into abstraction level eliminates the platform dependency and reduce the headache of the software developer. However this approach has several other advantages as well, one of which is we can understand the behavior of the system before actually implementing it and thus we can know the worth of that system[14]. When models are designed at abstraction level than number of transformation can be applied on that specific model to make it more and more platform specific model and the finally high level code of particular platform can be generated. With the help of this approach we can get different platform specific source codes from 1 model[15]2.

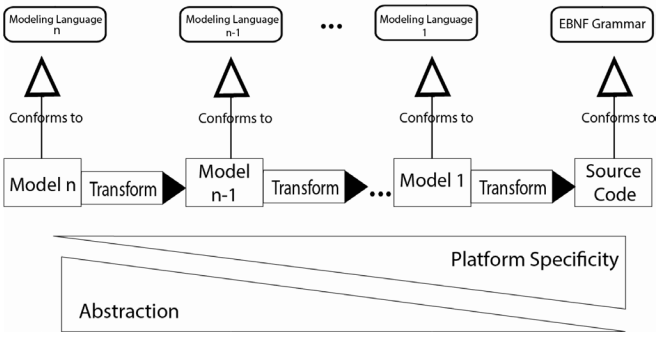
Model-Driven Engineering

Longtime before a de-facto standard was released by Object Management Group(OMG Group) which was entitled as "Model Driven Architecture" [4]. The duty of this standard is to separate the different components based on their platform dependency level [5]. MDA is architecture containing all the components of the Model driven paradigm or in other words, Model-driven Architecture is a super set of all model-driven paradigms. Such as Model-Driven Engineering (MDE), Model-Driven Software Engineering (MDSE), Model-Driven Development (MDD) and Models as well[16].

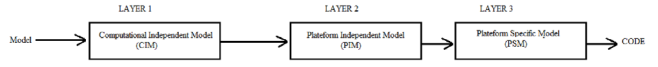
MDE is a technique of software engineering in which models are considered as main building blocks, unlike other software engineering approaches in which programs are the main artifacts [1]. In the MDE first desired system is first abstracted into the model and then the number of transformations is applied into the model making it more and more platform specified and the finally into machine code of the particular

Virtual University of Pakistan[1], Sindh Agriculture University Tandojam[2, 3, 4]
Email: azharkhaskheli@gmail.com

platform[2]. This behavior of model-driven engineering can be seen in the following figure[3], that how a model, let's call it 'n', is transformed from abstraction level into platform specified level, call it 'Model 1' through a number of iterations and finally source code[3].



Liviu defined MDE as a [5] three-layer process of transformation of model. At the first layer, the mode is which Computational Independent Model(CIM) is transformed to Platform Independent Model(PIM) which is then transformed into Platform-specific Model(PSM) in Layer three and finally source code is generated of a specific platform[17].



Model-Driven Development (MDD) comes under the umbrella of Model-Driven Software Engineering and has a very limited scope. Model-Driven Development is limited to layer 3 of Liviu 3 layer approach [5], though it is only responsible for the transformation from Model to Code.

Besides all the discussion of Model Driven Architecture and its comprising components, Model is the focal point in all. The model is an abstract representation of any real-world entity/ System. It must be clear and well defined if it is not then it may create problems at later stages. There are 3 main properties of the model that it must match[15].

- A Model is a representation of a system, though it should be based on something that is already built or is going to built or it should be completely imaginary[5].
- Model is a reduced version of the system, though it should not contain all properties it has to contain some properties[5].
- A Model always has a relation with the subject though it should be usable in place of the subject[5].

Modeling Languages

Models are designed using Modelling Languages, there is a number of modeling languages used for this purpose, some of them are general-purpose modeling languages while others are Domain Specific Languages (DSL)[17].

Domain-specific languages are those languages which are designed by domain experts to capture their requirement and design the model of their desired domain, an example of such domain specific languages include[2]

- Eclipse Modeling Framework(EMF)[6]
- Eclipse Based Unified Modelling Language Tools[7]
- Magic Draw[8]
- System Modeling Languages (SysML)[9]

Among all these tools EMF is very popular and widely used because of its simplicity and ease of use. The very most popular example of General Purpose Modelling Languages includes Unified Modelling Language(UML) which has become the de-facto standard of modeling language by Object Management Group in [12]. UML is used to specify, construct, visualizes, analyze, and document any project development or a software system. The UML diagrams help a developer to create a robust and more secure system in execution. Or UML well known for, creating objectoriented designs, diagrams, models, or documentation models.

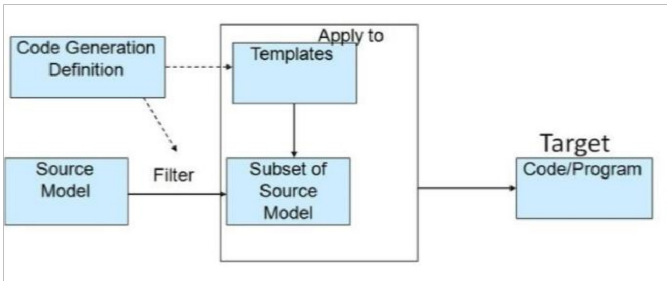
Model Transformations

Model Transformation is an operation on models that automatically generates a new model from our selected model according to some ruled defined [10]. There can be a number of operations or transformations such as Model to Model Transformation, Model to Text Transformation, etc.

Model to Model transformation is done in order to reduce the abstraction levels of model to make it platform-specific, This activity is limited to Liyiu's[5] first three layers of layers approach i.e. CIM to PSM. Model to Text Transformation is an approach in which graphical model is converted into textual representation, usually that textually representation is the source code of the particular programming language, this transformation simply transforms the PSM model into Code. Text to Model Representation is the reverse process of Model to text, this converts textural representation back to model representation[13].

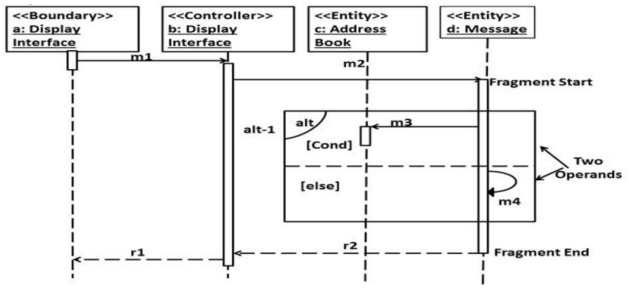
Code Generation

Code Generation is a type of transformation in which the model is transformed into textual representation, Code generators are responsible for these transformations. Code Generators are programs that create other programs. This complete process of code generation should be performed semi-automatic to increase its accuracy. It's a good practice that general-purpose parts which are common/standard should be generated using code generator while domain-specific parts should be made by developers[12].

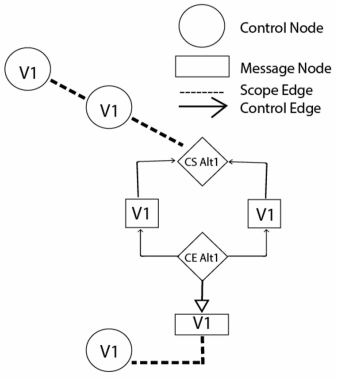


The behavior of code generation can be seen in the above figure. Code Generation approaches inputs the subset of source model which is our desired model to be transformed into the target which is source code or program, code generation uses code generation definitions or rules which helps code generator to transform the source model into target[13]

Kundu, Samanta, and Mall[12] defined a technique to transform a UML Sequence Diagram into a source code. Below is an example of a sequence diagram that is being transformed[12].



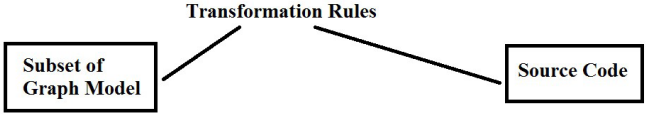
The graph model of this sequence diagram can be shown in figure.5. There is conditional flow in sequence diagram which means that if the condition is satisfied then the left side of the graph model will be executed otherwise right[1].



Source code can be generated from above graph model or a subset of that model by the help of code generator which uses transformation rules, those rules are nothing but they just define how the different component of the graph should be transformed into source code. An example of transformation rules can be seen in the following table[2].

Rule	M.Element	Branch	CA
1	C^{S}_{loop}	-	While(
2	V_i<SO, RO, RC, MT, PR, Rvar>	-	Rvar= RO.MT(PR);
3	C^{E}_{loop}	-	}
4	C^{S}_{alt}	-	If(
5	C^{S}_{alt}	Left	Else If(
6	C^{S}_{alt}	Middle	Else
7	V_i<SO, RO, RC, MT, PR, Rvar>	Right	Rvar= RO.MT(PR);
..	..	..	..

The Transformation rules contain a number of rules which include graphical Notation represented by Model Element and their corresponding Code. When the above transformation rule is used with a subset of our sequence diagram graph model below source code is generated [12].



Target Code

```
Class Bookregisgter{ FindBook(BookID){
        while(i<BookList.length & found=F){
                found=BookList[i].Match(BookId);
                }
         if(found==T){
        return BookList[i];
                }
         else{
        return null;
                }
        }
}
```

CONCLUSION

MDA is the father of all model-driven paradigms, among all model is the key element which is considered to be a very fundamental building block, that model can be designed using modeling languages those modeling languages include domain-specific modeling language and generalpurpose modeling languages. Unified Modelling Language is considered to be the de-facto standard of modeling language which is standardized by OMG and used by many companies including, Microsoft, IBM, etc. there can be a number of operations that can be applied on model among which model transformation is very useful which is used to convert the model from one representation into another. This transformation can be applied in the form of code generation which is used to transform the subset of a model into the desired source code. In this paper, we saw how different modeling paradigms relate to each other and how a UML sequence diagram is converted into source code.

REFERENCE

[1] Ali, A., Koondhar, M. Y., Depar, M. H., Maher, Z. A., Rind, M. M., & Shah, A. (2021). Framework for Location Based Attendance System by Using Fourth Industrial Revolution (4IR) Technologies. International Journal, 10(3).

[2] Khaskheli, S. A., Pathan, M., Qureshi, B., Khaskheli, A. A., Ahmed, T., Nizamani, N. N. D., & Dahri, F. (2021). AI Based Motor Vehicles Detection and Tracking System Using Smartphone Application. International Journal, 10(3).

[3] Gurunule, D., & Nashipudimath, M. (2015). A review: analysis of aspect orientation and model driven engineering for code generation. Procedia Computer Science, 45, 852-861.

[4] Staab S., Walter T., Gröner G., & Parreiras, F. S. Model driven engineering with ontology technologies. Reasoning Web. Semantic Technologies for Software Engineering. Springer Berlin Heidelberg; 2010.p. 62-98.

[5] OMG. MDA Guide version 1.0.1. OMG document omg/2003-06-01, 2003

[6] Cretu, L. G. (2014). Model Driven Engineering Using UML. A Pragmatic Approach. Model Driven Engineering of Information Systems: Principles, Techniques, and Practice, 229.

[7] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. 2008. EMF: eclipse modeling framework. Pearson Education.

[8] Agnes Lanusse, Yann Tanguy, Huascar Espinoza, Chokri Mraidha, Sebastien Gerard, Patrick Tessier, Remi Schneken- burger, Hubert Dubois, and François Terrier. 2009. Papyrus UML: an open source toolset for MDA. In Proc. of the Fifth European Conference on Model-Driven Architecture Foundations and Applications (ECMDA-FA 2009). Citeseer, 1ś4.

[9] Magic, N. (2007). MagicDraw. URL https://www. nomagic. com/products/magicdraw.

[10] Sparx Systems. 2012. Sparx Systems SysML. https://sparxsystems.com/. (2012). [Accessed: 23/03/2020].

[11] Kleppe, Warmer J, Bast W. MDA Explained, The Model-Driven Architecture: Practice and Promise. Addison Wesley; 2003.

[12] Kundu, Debasish, Debasis S, and Rajib M. Automatic code generation from unified modelling language sequence diagrams. Software, IET7.1; 2013.p. 12-28

[13] Knapp, A., & Merz, S. (2002). Model checking and code generation for UML state machines and collaborations. Proc. 5th Wsh. Tools for System Design and Verification, 59-64.

[14] España, S., Bik, N., & Overbeek, S. (2019, May). Model-driven engineering support for social and environmental accounting. In 2019 13th International Conference on Research Challenges in Information Science (RCIS) (pp. 1-12). IEEE.

[15] Verbruggen, C., & Snoeck, M. (2023). Practitioners' experiences with model-driven engineering: a meta-review. Software and Systems Modeling, 22(1), 111-129.

[16] Mohamed, M. A., Challenger, M., & Kardas, G. (2020). Applications of model-driven engineering in cyber-physical systems: A systematic mapping study. Journal of computer languages, 59, 100972

[17] Chillón, A. H., Ruiz, D. S., Molina, J. G., & Morales, S. F. (2019). A model-driven approach to generate schemas for object-document mappers. Ieee Access, 7, 59126-59142.