Journal of Theoretical and Applied Information Technology
30th November 2015. Vol.81. No.3
© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

MULTI-CRITERIA ANALYSIS AND ADVANCED COMPARATIVE STUDY BETWEEN AUTOMATIC GENERATION APPROACHES IN SOFTWARE ENGINEERING

[1] ZOUHAIR IBN BATOUTA, [2,1] RACHID DEHBI, [1] MOHAMMED TALEA, [1] OMAR HAJOUI
[1]Hassan II University, Faculty of Science Ben M'Sik, LTI Laboratory, MOROCCO
[2]Hassan II University, Faculty of Science Aïn Chock, LIAD Laboratory, MOROCCO
E-mail: zouhair.ibnbatouta@gmail.com , dehbirac@yahoo.fr, taleamohamed@yahoo.fr , hajouio@yahoo.fr

ABSTRACT

New development methods have emerged in recent years. These techniques are based on models and software components, they aim to facilitate integration, automation, and generation of complex applications, as well as mapping between different platforms based on forward and reverse engineering. These approaches are based on Model Driven Engineering (MDE) which separates the business logic of an application from the technology used to achieve it. This paper aims to provide a best understanding of MDE aspects and presents a comparative study between different approaches of software development automation and code generation, in addition a big contribution of this article is to present the forces and weaknesses of each approach based on a multi-criteria analysis method, this is our first step to design and implement concrete and effective solutions for automatic generation issues, Moreover, this study will also help the professionals in decision-making by facilitating the choice of The best approach to be used according to desired criteria and their importance. Our article goes into a global objective that aims automating the generation.

Keywords: *Model Driven Engineering (MDE), Forward Engineering, Reverse Engineering, Software Development Automation, Code generation.*

1. INTRODUCTION

Nowadays, computer applications are more important in daily living, these applications have become increasingly complex, and so is their realization. In addition, Contracting Owners are more and more demanding in terms of quality, cost and delivery time. Another aspect that can engender many problems, is the diversity of used platforms for implementation [1]-[5], which require the intervention of more and more experts in business, functional and technical fields.

In order to facilitate application development, software engineering has greatly improved over the last few years, moving from procedural technology towards the Object-oriented technology in the 70s, and Components Oriented technology in the 90s [2]. However, the main software engineering problems couldn't have been overcome, namely interoperability issues, development and migration costs, delivery deadlines issues. Thus, it is necessary to adopt the approach of model driven engineering in lieu of code driven engineering [3].

Model Driven Engineering emphasizes some more important aspects of models. Indeed, in this approach, the model goes from a simple contemplative vision, whose aim is to improve the documentation and specifications, towards a productive vision regarding final code generation for a given platform [4].

Even-though MDE has made some significant contributions to today's world software engineering, there are still numerous challenges that need to be profoundly addressed [5]. In this article we present a detailed comparative study between different approaches, used in the automatic applications generation. We will start by describing model driven engineering basic languages and standards. After that, we will define each generation approach apart and the standards on which it is based. Then, we will carry out a comparative study between the different approaches, explaining in detail the advantages and disadvantages of each one of them.

Next, we will establish a conformity criteria repository to test the different approaches.

2. MODEL DRIVEN ENGINEERING

2.1 Definition

Model driven engineering appeared as early as 2000 as an improvement to code driven engineering [2]. It sets the model at the heart of the development process in order to facilitate automatic processing through the models reuse and transformation. The model goes from a static and contemplative vision to a productive vision (Figure 1) [4].

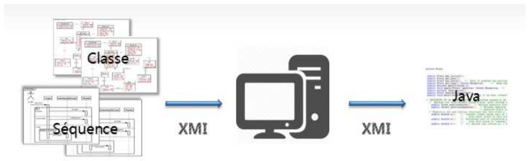


Fig. 1. Evolution From The Contemplative Vision/Productive

Model Driven Engineering (MDE) advocates to support the well known principle of separation of concerns through the extensive use of models in all the steps of the software development cycle [6]-[7]-[8]. Indeed, it distinguishes the business layer of the implementation platform to automate and facilitate the transformation process. We will immediately explain some languages and basic standards for model driven engineering.

2.2 Basic MDE Languages And Standards

- Metamodel: or surrogate model is a model of a model, each model must comply with its metamodel.
- MODEL Transformation Language: it is a language that allows model transformation. Examples : ATL, GReAT, JTL, Kermeta, Lx family, Object Management Group (OMG) standard QVT, M2M Eclipse based on QVT standard [9]-[12].
- Model-to-text transformation languages: allows model to code transformation. Exemples : MOFM2T based on QVT standard, M2T Eclipse, Epsilon Generation Language (EGL) [10]-[12]
- Domain specific Language (DSL): allows to create metamodels. Examples: MOF: OMG, JMI Java API for manipulating MOF models.[9]-[10]-[12]-[25]
- XML Metadata Interchange (XMI): the OMG standard that allows model conversion to XML [9]-[10]-[12].
- The Web Ontology Language (OWL) which is a semantic markup language for publishing and sharing ontologies on the World Wide Web [11]-[12].
- The Common Warehouse MetaModel (CWM): it standardizes a complete, comprehensive metamodel that enables data mining [12].

3. MDE APPROACHES

In this section, we will see the different methods that exist in the MDE and allow the automatic or semi-automatic generation of applications. These methods are classified in three major approaches: the generative approach, the interpretive approach and the hybrid approach [13]-[14]-[15]. We will see the definition and standards of each approach, as well as the existing tools in the market that are based on each one, after that, we will make a comparative study and Multi-criteria analysis between these approaches.

3.1 Generative Approach

3.1.1 Definition

The generative approach is essentially based on the transformation of high-level abstraction models to low-level abstraction models or possibly to the code. This approach involves taking multiple input models to turn them into final code through successive transformations that may use specific models for these transformations [13]-[14]-[15].

This method is widely used in the MDE, we can take for example LMS Generator which is a generation system for eLearning platforms [35]-[36]-[38]. Figure 2 shows the different steps of this approach, starting with high level models that can be refined and transformed to lower-level models, with the possibility of reverse engineering to rectify the upstream models.

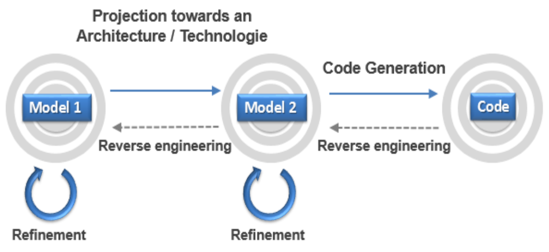


Fig 2: MDD Generative Approach

In the next section, we will explain the most used standards for the current approach.

3.1.2 Standards

In this section, we will see three very well-known and used standards in this approach. Namely, the OMG MDA standard, the OPM standard and finally the Microsoft Domain-Specific Language.

3.1.2.1 MDA standard

MDA is a software development lifecycle method that was introduced by the Object Management Group (OMG) in 2001. MDA is among the essential standards used by the tools adopting the generative approach, it is based on the separation of concerns concept.

The main idea behind MDA is to use models as core development artifacts and thus be able to separate platform specific data from the software development process. Developing applications without platform specific terms makes it easier and less costly to port them to different platforms [16]. This makes easier the multi-target code generation: write once, run everywhere; model once, generate everywhere [17].

A. MDA development process:

The two main artifacts of MDA are models and models transformation, we can distinguish four important models:

- Computation Independent Model CIM: It is the first model of the MDA approach, it allows specifying and modeling customer needs. Despite the importance of this model, it is not always taken into account by the tools using the MDA approach [16]-[37].
- Platform-Independent Model PIM: This is one of the major MDA models. It allows the separation of the application business aspect from the implementation platforms, in order to facilitate the generation of the application in different target platforms with a minimal cost [16]-[37].
- Platform Specific Model PSM: This is the second important model of the MDA approach. It follows from the PIM transformation. This model is related to an execution platform, it is the closest low-level model to code and can easily be converted to the corresponding platform code [16]-[37].
- Platform Description Model PDM: It contains information for models' transformation to a platform. Basically, it allows the passage from PIM to PSM. This model should normally be delivered by the platform builder to facilitate this transformation [16]-[37].

Figure fig3 explains the process of making an application following the MDA standard:

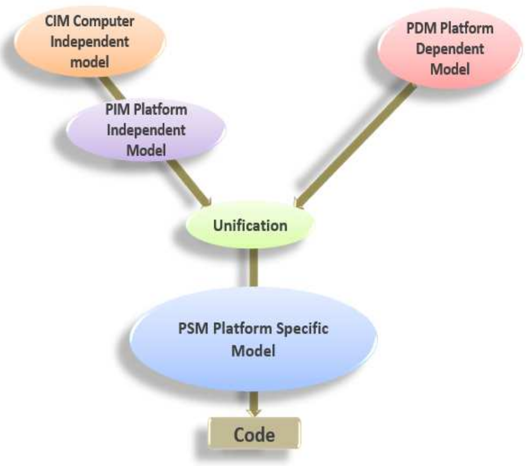

CIM Computer Independent model
PDM Platform Dependent Model
PIM Platform Independent Model
Unification
PSM Platform Specific Model
Code

Fig 3: Transformation Process According To MDA

Figure fig4 shows an example of models utilization in order to realize an application:

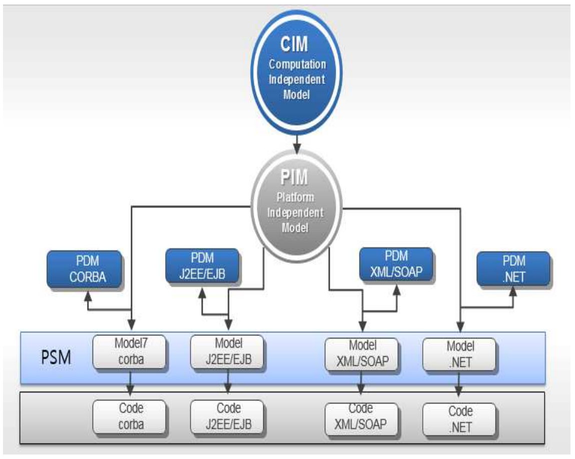


Fig 4: Example Of Using Models To Realize An Application

B. MDA 4-layer architecture:

In the OMG proposed approach “MDA”, a multi layer architecture was defined, it is called “four-layered architecture”. It is based on the following concepts: models, metamodels and meta-metamodels. The layers in that architecture are called M0, M1, M2 and M3. Each layer depends on the layer above and represents somehow its instance, except for the layer M0, which does not depend on any layer, since it is the highest layer.

M0: it represents the real world, Layer M0 specifies user objects that are instances of the UML user model classes [17]-[18].

M1: This layer is based on elements, which represent models. An example would be a UML model of a software system. M1 layer is a model of the M0 layer user data [17]-[19]-[20].

M2: It contains models of layer M1 models. M2 models are known as metamodels [17].

M3: It contains models of layer M2 models. M2 elements are known as meta-metamodels.MOF is the standard for defining the layer M3 elements [17]-[19].

Figure fig5 presents a 4 levels architecture for the MDA approach:

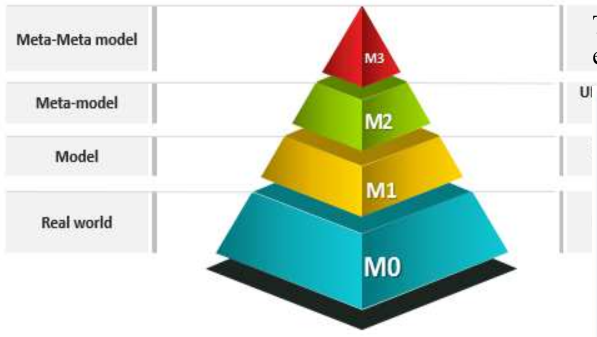


Fig 5: MDA 4 Levels Architecture

C. UML profiles:

A UML profile is a specification of a UML model, it provides a generic extension mechanism for customizing UML models for particular domains and platforms [21]. UML Profiles are widely used in the MDA approach, especially for PSM models that depend on the execution platforms like J2EE or .Net [22]. The figure fig6 shows an example of profiles use in the MDA approach:

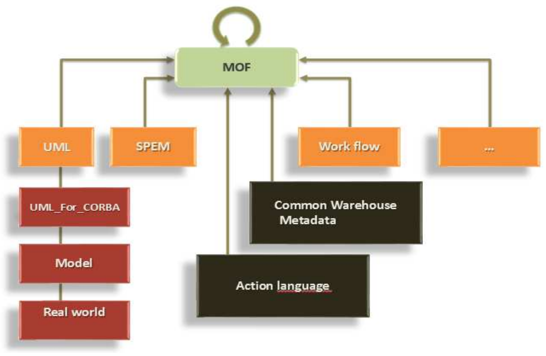


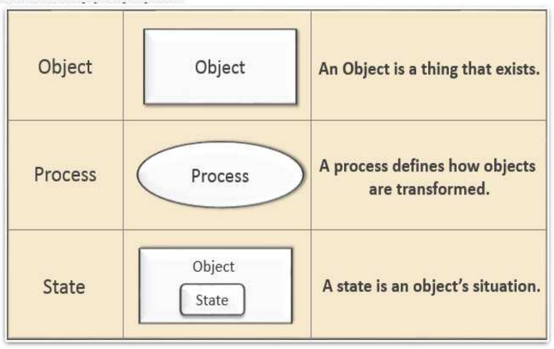
Fig 6: UML Profile For MDA

3.1.2.2 OPM standard

The OPM methodology (Object Process Methodology) combines two main and important concepts, namely the object and process. It is an extension of the object-oriented design based on objects. The ability to unify the object and the process in a single model made this approach a robust and reliable method to use [14]-[23].

This method contains two main aspects, namely OPD diagrams (Object-Process Diagrams), which are graphical models, and the OPL, which is the textual writing language equivalent to these models. Graphic models are well organized and follow a well-defined design. They are made up of entities, fundamental structural relations and procedural links. The OPL is very a strong language, indeed, it can be read by humans and at the same time it can be interpreted by machines, consequently it is an inter-exchangeable language [23]-[24].

Table tab1 shows the OPM method essential entities:



Object	Object	An Object is a thing that exists.
Process	Process	A process defines how objects are transformed.
State	Object State	A state is an object's situation.

Tab 1: OPM Entities

3.1.2.3 Microsoft domain-specific language

Another variant of the IDM is the DSL tools from Microsoft that allows to create specific DSL, in order to facilitate code generation, Microsoft DSL [25]:

- Gives the possibility to work on a domain specific language via a graphical designer (serialized in a proprietary XML format). Thus, one can define and edit those languages.
- Allows setting designer definitions via a proprietary XML format, this format is the source to generate the code (without any manual intervention), which implements graphical modelers of the DSL.
- Includes Code Generators that take a DSL definition and a designer producing the

code that implements the graphical editors [25].

- Includes a framework to define code generators based on template languages that take an instance of a domain model and generates code based on the template [25].

3.2 Interpretive Approach

3.2.1 Definition

The interpretive approach is different from the generative approach because it does not generate the final code, it uses directly interpretable models, so there is no need of an intermediary execution language to interpret the application, thus the model is considered as interpretable code. Tools based on this approach generally combine several executable models to launch the application, and that using an adequate internal interpreter. In the next section, we will see an important standard for this approach, namely the OPG (Open Process Graph) [13]-[14]-[15]-[27].

3.2.2 OPG standard

The Object process graph incorporates the concept of a graph-oriented object database model. It is mainly based on models direct interpretation over three essential aspects; namely the process aspect that represents the application business logic, the user interface aspect which represents the client and end user view, and finally the database aspect that allows the application data storage [27].

Direct interpretation of this graph is used to start the application without needing the intermediate code. Tools using this approach usually have a powerful interpreter for combining the aforementioned three aspects, in order to perform the application execution.

The OPG incorporates various elements of the object-oriented approaches in order to represent the business aspect of the application, such as classes, methods, various relational databases elements as well as various graphical controls required for user interface development [27].

Figure fig6 shows a diagram explaining the OPG process.

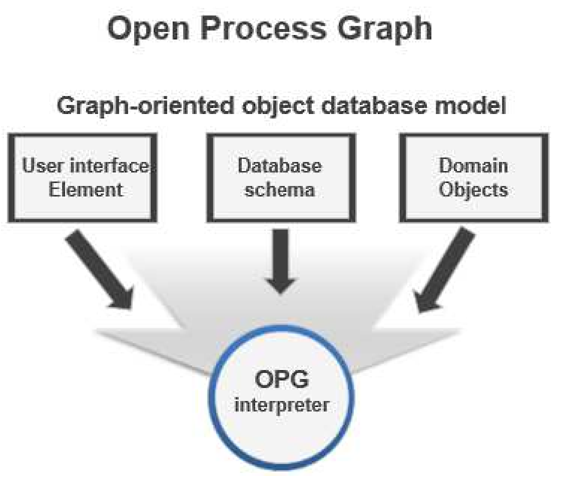


Fig 6: OPG Diagram

3.2.3 Executable UML

Executable UML is a software development technique based on the concept of domain or aspect. In Executable UML, a system is a set of domains which represent its subject matters. Executable UML allows to model a domain in the level of abstraction of its subject matter, in an independent way of any implementation concern [28].

The obtained domain model is composed of four elements. The first element is the domain chart, which offers a view of the modeled domain and its dependencies with other domains. The second one is the class diagram, it provides the definitions of the domain classes and their associations. The third element is the statechart diagram, this latter provides the following definitions for the classes or their instances: the states, the events and the state transitions. The last element is the action language, this one aims to define actions or operations which apply to model elements in order to process them [28].

Executable UML can be used either as an executable code or as documentation.

3.3 Hybrid Approach

A new approach has emerged combining both generative and interpretive approaches. This approach tries to find a balance between these two methodologies. For example, the model interpreter can be enriched by generated code, and that in order to facilitate interpretation [13]-[14]-[15], without having to generate all the application code, the model keeps a place in the final application and can act directly at runtime. Figure fig7 shows an example of this combination of generated code and models.

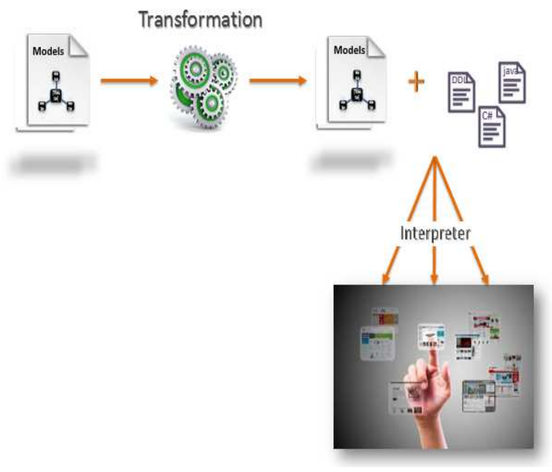
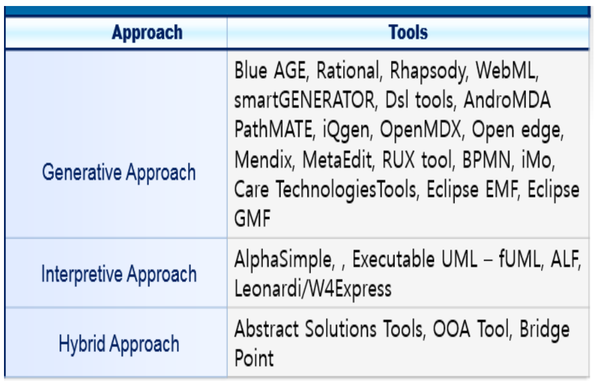


Fig 7: Hybrid Approach Explanatory Diagram

3.4 Tools

Table tab2 shows the commonly used generation tools on the market and the corresponding approaches:



Approach	Tools
Generative Approach	Blue AGE, Rational, Rhapsody, WebML, smartGENERATOR, Dsl tools, AndroMDA PathMATE, iQgen, OpenMDX, Open edge, Mendix, MetaEdit, RUX tool, BPMN, iMo, Care TechnologiesTools, Eclipse EMF, Eclipse GMF
Interpretive Approach	AlphaSimple, , Executable UML – fUML, ALF, Leonardi/W4Express
Hybrid Approach	Abstract Solutions Tools, OOA Tool, Bridge Point

Tab2: Example Of Generation Tools With Their Approach

4. COMPARATIVE STUDY OF MDE APPROACHES

The existence of these three approaches of automation makes it difficult to choose among them, the objective of this section is to make an advanced comparative study between these three concepts. This study will be performed in three parts, the first containing the advantages of each approach, the second contains the downsides and the last contains a summary of the comparison according to well-defined criteria.

4.1 Advantages

4.1.1 Generative approach

The generative approach is widely used in automation, due to the fact that compared to the interpretative approach, it generates interpretable code depending on a given platform which has several advantages. Among the generative method strengths, we can name:

- Code generation to the desired platform: Indeed the tools based on this approach are generally used to transform the high level models to several types of platforms as needed.
- Separation of development environments: This approach allows to separate the modeling environment dedicated to the development and model transformations, from the execution environment dedicated to the interpretation of the final code. This gives developers a second degree of freedom since they can act on the models or on the generated code.
- Faster Execution time: The generated code in a given platform is easier to interpret than a model or a set of high level models or even of higher level, since the code level is the lowest model that can be generated. It allows faster execution and therefore a very significant time saving compared to the performance of models execution.
- Optimized applications in the side of targeted software: Since the targeted software of transformations is chosen from the beginning if necessary, it helps to better optimize the generated code in both functions and database storage. This leads to more reliable and better applications.
- Modernization and Reverse Engineering easier: The software modernization is very important, given the continuous evolution of development and execution platforms. Reverse engineering is the process allowing to move from a final application to a model or a set of specific models, which can also be described by the word transmodeling. Having an existing code facilitates the transmodeling to the closest low level models, which optimizes the process of modernization.
- Higher models security: The models are not exposable given that the interpreter needs only the generated code, it allows great protection to models that are independent of the execution platform, and therefore isolated to intrusion attacks.

4.1.2 Interpretive approach

The interpretive approach is also very present in the generation and automation tools, it has several strengths that make more and more software use it. Among the interpretive method strengths, we can mention:

- Simplified process: In the absence of a final generated code, the model becomes more important in this approach, it is directly executed by the interpreter, and therefore the development process becomes reduced and simpler.
- Easy implementation and use of the environment: The development environment is easy to implement since there is no multitude of generation target areas. The tools use their own technologies to directly interpret the different aspects of the used models, be it business or database.
- Reduced development time: This is due to the minimum number of transformations between models, the nonexistence of high-level model transformations to low level ones or to the code patterns which take a considerable time.
- Directly changing the visible model in the interpretation: This simplifies the correction of errors and models optimization, no need to go through a second execution platform or the generated code to detect coding issues.

4.1.3 Hybrid approach

The hybrid approach try to takes a bit from the advantages of both approaches, among the hybrid approach strengths, we can name:

- Semi generation de code: even if the model can act directly at runtime, a generated code is added to facilitate the process of interpretation.
- Reasonable interpretation time: code artifact belong to a lower layer so it reduce the time of execution even if the model involvement can limit a bit this faculty.
- Shorter process: the presence of both executive models and generative code make the process lighter since it is not necessary to transform all the models into code artifact.
- Reasonable development time: it is not necessary to transform all models to code layer which decrease the time of realization.

4.2 Disadvantages

4.2.1 Generative approach

The generative approach uses a large number of models and therefore more transformations are required between high-level models and low-level ones. This makes the development process heavier and increases the time of applications realization.

The diversity of the possible target code generation platforms makes the development more complex. Developers must be experts and have global notions about the different possible programming languages.

4.2.2 Interpretive approach

The interpretative approach has several disadvantages. Among them, we can name, fewer degrees of freedom, the use of platforms set by the generation tool provider, which makes the effectiveness of the generated implementation questionable since the used target tools are not always optimal.

Another disadvantage is the execution time of the resulting solution, which is longer because the models are relatively more difficult to interpret than the executable code. Another weakness of this approach is the models security since these models are exposable because these models are considered as a code, and thus vulnerable to attacks. The modernization also becomes more difficult as reverse engineering is heavier because it is always more challenging to go from a final solution to a complex high level model.

4.2.3 Hybrid approach

The hybrid approach has several weaknesses, among them:

- Limited use of platforms: the developer must necessarily use the templates and target languages imposed by the provider of the solution.
- Exposed models: models are partly exposed due to their importance in execution and therefore more difficult to secure.
- Slow modernization: the mixture between the model and the code generate makes the more difficult the transmodeling as well as the reverse engineering.

- Longer execution time: the execution procedure shall combine the results of interpretation of models and those of the generated code and as a result the slowness in the implementation.

4.3 SWOT Analysis

In this section we present a SWOT analysis summarizes the strengths and weaknesses of these approaches:

Generative	
Positives	• Code generation to the desired platform • Separation of development environments • Faster Execution time • Optimized applications in the side of targeted software • Modernization and Reverse Engineering easier • Higher models security
Negatives	• large number of models • more transformations are required • heavy development process • big time of applications realization • the development more complex • High level of expertise required
Interpretive	
Positives	• Simplified process • Easy implementation and use of the environment • Reduced development time • Directly changing the visible model in the interpretation
Negatives	• fewer degrees of freedom • Limited use of platforms • target tools are not always optimal • Longer execution time • More difficult interpretation • Models are vulnerable to attacks • More difficult modernization • Reverse engineering is heavier
Hybrid	
Positives	• Semi generation de code • Reasonable interpretation time • Shorter process • Reasonable development time
Negatives	• Limited use of platforms • Exposed models • Slow modernization • Longer execution time

Tab3: SWOT Analysis

4.3 Multi-criteria comparative study:

4.3.1 Multi-criteria analysis:

After seeing the advantages and disadvantages of each approach, we will now develop a multi-criteria analysis between these approaches. Multi-Criteria Decision Analysis, or MCDA, is a valuable tool that can be applied to many complex decisions. It can solve complex problems that Include qualitative and/or quantitative aspects in a decision-making process.
The score of an approach is calculated based on a number of criteria. So far we have identified ten ; Indeed, based on the characteristics of each of the approaches presented in the comparative study and the SWOT analysis presented by the front, we have determined ten important criteria: Adaptability, Required skills, Execution time, Development time, Bases and standards, Generation Simplicity, Fields of application, Integration of new generators, Models' Security, Reverse Engineering.

4.3.2 Multi-criteria analysis method:

To make the comparison between the approaches using a number of criteria, there are several possible mathematical methods. These methods can be divided into three main families [30]-[31]-[32]-[33]-[34]:

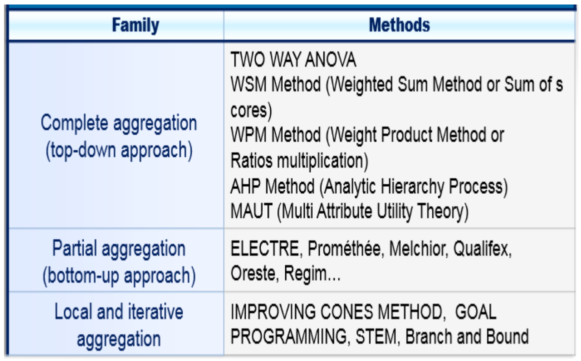
- **Complete aggregation (top-down approach):** This approach seeks to aggregate the n criteria to reduce them to a single criterion.

- **Partial aggregation (bottom-up approach):** This approach seeks to compare potential actions or rankings to each other, and to establish between them outranking relations.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- **Local and iterative aggregation:** This approach looks primarily for a starting solution. Thereafter, we proceed to an iterative search to find a better solution.

The table tab3 shows the different existing multi-criteria methods sorted by family [30]-[31]-[32]-[33]-[34]:



Family	Methods
Complete aggregation (top-down approach)	TWO WAY ANOVA WSM Method (Weighted Sum Method or Sum of s cores) WPM Method (Weight Product Method or Ratios multiplication) AHP Method (Analytic Hierarchy Process) MAUT (Multi Attribute Utility Theory)
Partial aggregation (bottom-up approach)	ELECTRE, Prométhée, Melchior, Qualifex, Oreste, Regim...
Local and iterative aggregation	IMPROVING CONES METHOD, GOAL PROGRAMMING, STEM, Branch and Bound

Tab4: Example Of Multi-Criteria Analysis Methods

4.3.3 Weighted Sum Method (WSM):

For our analysis, we chose the Weight Sum Method (WSM). Indeed, this method allows to find the best possible approach by assigning a weight to each comparison criterion, it allows to take into account all the criteria according to their value and without a criterion penalizing the other criteria [30]-[31]-[32]-[33]-[34].

This method is based on five key elements:

- Potential n actions set

$A=\{a_1,a_2,a_3,\ldots,a_n\}$ a_i, where $i=1,2,\ldots,n$

- M different criteria c_j where $j=1,2,\ldots,m$
- Criteria weights p_j for each criteria where $j=1,2,\ldots,m$
- Evaluations or judgments e_{ij} for each action on each criteria where $i=1,2,\ldots,n$, $j=1,2,\ldots,m$
- max or min $\sum e_{ij}*p_j$ for $i=1,2,\ldots,n$, in our case we need to maximize this function to have the better solution

4.3.4 Comparison *criteria and weight*

We present in this chapter the ten comparison criteria cited on which the comparative study will be based, we notice that these criteria are based on the characteristics of each of the approaches presented in the comparative study and the SWOT analysis presented by the front, we summarized all the characteristics (strengths and weaknesses) in ten global criteria to ensure better analysis and optimize the comparison, these criteria are:

- **C1 : Adaptability :** It is the power to adapt to different platforms
- **C2 : Required skills :** The larger the required skills are, the more complex and exploitable the approach is
- **C3 : Execution time :** The running time is critical in judging of the effectiveness of an approach.
- **C4 : Development time :** the more efficient the approach is, the more the development time is reduced
- **C5 : Bases and standards :** It is paramount that the approach is based on international standards, the more known the standards are, the more effective the approach is.
- **C6 : Generation Simplicity:** The generation process must be as short as possible and as effective as possible
- **C7 : Fields of application:** This criterion is used to verify whether the approach is used in different fields (web, mobile application, cloud...) or just for a reduced domain.
- **C8 : Integration of new generators:** This criterion is very important because of continuous progress of development platforms, it is therefore essential that the approach ensures the integration of application generators to new platforms.
- **C9 : Models' Security:** The more the model is exposed, the less it is secure.
- **C10 : Reverse Engineering :** The approach that facilitates reverse engineering is favored over other approaches, due to the fact that an important aspect of the MDE is the modernization of legacy applications.

These criteria are classified according to the following order of importance:

Adaptability = Reverse engineering > Development time = Generation Simplicity = Field of application = Integration of new generators = models' security > Bases and standards = Execution time = Required skills

And therefore the WSM weight accorded are as represented in tab4 below:

Criteria	Weight
Adaptability - Reverse engineering	4
Development time - Generation Simplicity - Field of application - Integration of new generators - models' security	3
Bases and standards - Execution time - Required skills	2

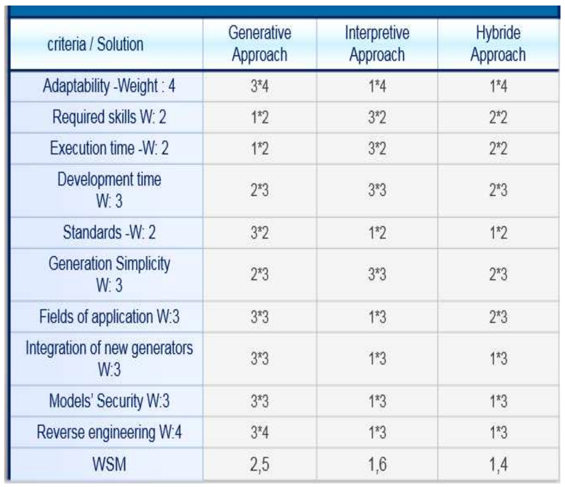
Tab5: Criteria Weight

4.3.5 Multi-criteria choice matrix:

The first step in applying the WSM method is the carrying out of the multi-criteria choice matrix. This matrix' columns contain the approaches to be compared and its lines contain the different criteria with the weight assigned to each criterion according to its importance. In cells there is the score given to each criterion approach based on the detailed comparative study of the approaches, that score can have three values: 3 meaning GOOD, 2 meaning MEDIUM and 1 meaning LOW [30]-[31]-[32]-[33]-[34]:

- 3: means that the intended approach is good for the given criterion.
- 2: means that the intended approach is average for the given criterion.
- 1: means that the intended approach is low for the given criterion.

Table TAB4 represents the resulting multi-criteria choice matrix:



criteria / Solution	Generative Approach	Interpretive Approach	Hybride Approach
Adaptability -Weight : 4	3*4	1*4	1*4
Required skills W: 2	1*2	3*2	2*2
Execution time -W: 2	1*2	3*2	2*2
Development time W: 3	2*3	3*3	2*3
Standards -W: 2	3*2	1*2	1*2
Generation Simplicity W: 3	2*3	3*3	2*3
Fields of application W:3	3*3	1*3	2*3
Integration of new generators W:3	3*3	1*3	1*3
Models' Security W:3	3*3	1*3	1*3
Reverse engineering W:4	3*4	1*3	1*3
WSM	2,5	1,6	1,4

Tab6: Multi-Criteria Choice Matrix

4.3.6 Curve and comparison histogram:

Figure Fig8 shows the distribution of the three curves representing the final scores for each approach against the comparison criteria.

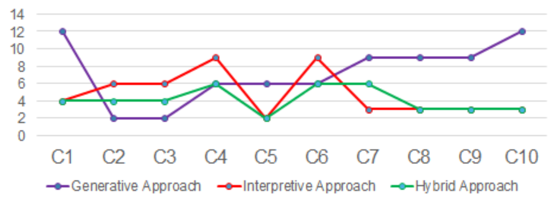


Fig8: Distribution Of Ratings Against The Criteria

The histogram in Figure Fig9 shows the final score for each approach. The best score obtained is 2.5 / 3, it shows a net benefit to the generative approach against the set of selected criteria and over the other approaches, it is followed by the interpretative approach, and the hybrid approach comes last. We can notice that none of these approaches could reach the perfect score 3/3 according to this comparative approach.

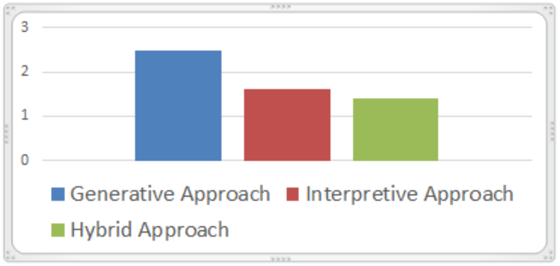


Fig9: Approaches Final Notation

5. CONCLUSON

The model-driven engineering plays a very important role in the generation of applications and simplifying the development process. This article has enabled us to understand all MDE aspects and usefulness, and to see in detail the approaches, namely the generative approach, interpretive and hybrid with the advantages and disadvantages of each one of them. We also carried out a detailed comparative study of these approaches in order to classify them according to defined criteria by using the WSM method, we have seen that these approaches still have many challenges, be it in the models' security, the detection of generation errors, complexity of use particularly for simple and non-complex applications. Another big challenge is the difficulty of integrating new code generators due to the rapid development of execution and deployment platforms, interoperability between MDE tools belonging to different approaches. As future work, we will present solutions for these issues. This work present this work gives a great contribution also for professionals to help them choose between different existing approaches, and this according to their needs and criteria that matter most to them, in addition our article goes into a global objective that aims automating the generation.

REFRENCES:

[1] KENT, Stuart. Model driven engineering. *Integrated formal methods, Springer Berlin Heidelberg*, 2002. p. 286-298.

[2] BÉZIVIN, Jean. From object composition to model transformation with the MDA, *tools. IEEE*, 2001. p. 0350.

[3] LEVEQUE, Thomas; ESTUBLIER, Jacky; VEGA, German. Extensibility and Modularity for Model Driven Engineering Environments, *Engineering of Computer Based Systems, 2009. ECBS 2009. 16th Annual IEEE International Conference and Workshop on the. IEEE*, 2009. p. 305-314.

[4] DE MIGUEL, Miguel; JOURDAN, Jean; SALICKI, Serge. Practical Experiences in the Application of MDA, *≪ UML≫ 2002—The Unified Modeling Language. Springer Berlin Heidelberg*, 2002. p. 128-139.

[5] García Díaz, V., Valdez, N., Rolando, E., Espada, J. P., Bustelo, P. G., Cristina, B and Montenegro Marín, C. E. (2014). A brief introduction to model-driven engineering. *Tecnura*, 18(40), 127-142.

[6] Etien, A., Muller, A., Legrand, T., and Blanc, X. (2010, March). Combining independent model transformations. *In Proceedings of the 2010 ACM Symposium on Applied Computing* (pp. 2237-2243).

[7] Miller. J., Mukerji. J Mda guide version 1.0.1. *Technical report, Object Management Group (OMG)* (2003)

[8] Selic. B, The pragmatics of model-driven development. *IEEE Software 20(5)* (2003) 19–25

[9] Czarnecki, K., & Helsen, S. (2003, October). Classification of model transformation approaches. *In Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture* (Vol. 45, No. 3, pp. 1-17).

[10] MOF Model to Text Transformation Language, v1.0 *OMG Available Specification* (http://www.lifl.fr/~dumoulin/enseign/pje/docs/MTL-08-01-16.pdf)

[11] Djurić, D., Gašević, D., & Devedžić, V. (2005). Ontology modeling and MDA. *Journal of Object technology*, 4(1), 109-128.

[12] MDA® Specifications, (*http://www.omg.org/mda/specs.htm#CWM)*

[13] Meijler, T. D., Nytun, J. P., Prinz, A., & Wortmann, H. (2010). Supporting fine-grained generative model-driven evolution. *Software & Systems Modeling*, 9(3), 403-424

[14] Nikola Tankovic, Model Driven Development Approaches: Comparison and Opportunities, (*https://www.fer.unizg.hr/_download/repository/N.Tankovic,_rad_za_KDI.pdf*)

[15] Tankovic, N., Vukotic, D., & Zagar, M. (2012, June). Rethinking model driven development: analysis and opportunities. *In Information Technology Interfaces (ITI), Proceedings of the ITI 2012 IEEE 34th International Conference on* (pp. 505-510).

[16] OMG. MDA Guide version 1.0.1, 2003. *OMG Document: omg*/2003-06-01.

[17] Chaouni, S. B., Fredj, M., & Mouline, S. (2011). MDA based-approach for UML Models Complete Comparison. *arXiv preprint arXiv*:1105.6128

[18] Bezivin, J. (2004, November). Model Engineering for Software Modernization. *In WCRE* (p. 4).

[19] Kleppe, A. G., Warmer, J. B., & Bast, W. (2003). MDA explained: the model driven architecture: practice and promise. *Addison-Wesley Professional*.

[20] Atkinson, C., & Kühne, T. (2003). Model-driven development: a metamodeling foundation. Software, *IEEE*, 20(5), 36-41.

[21] Rodrigues, A. W. D. O., Guyomarc'h, F., & Dekeyser, J. L. (2011). A modeling approach based on uml/marte for gpu architecture. *arXiv preprint arXiv*:1105.4424

[22] Fuentes-Fernández, L., & Vallecillo-Moreno, A. (2004). An introduction to UML profiles. *UML and Model Engineering*, 2

[23] Aharoni, A., & Reinhartz-Berger, I. (2008). A domain engineering approach for situational method engineering. *In Conceptual Modeling-ER* 2008 *Springer Berlin Heidelberg*. (pp. 455-468).

[24] Reinhartz-Berger, I., & Dori, D. (2004). Object-Process Methodology (OPM) vs. UML-a Code Generation Perspective. *In CAiSE Workshops* (1) (pp. 275-286).

[25] Pelechano, V., Albert, M., Muñoz, J., & Cetina, C. (2006, October). Building Tools for Model Driven Development. Comparing Microsoft DSL Tools and Eclipse Modeling Plug-ins. *In DSDM.*

[26] Van Deursen, A., Klint, P., & Visser, J. (2000). Domain-Specific Languages: An Annotated Bibliography. *Sigplan Notices,* 35(6), 26-36.

[27] Gold, S. A., Baker, D. M., Gusev, V., & Liang, H. (2008). U.S. Patent No. 7,316,001. *Washington, DC: U.S. Patent and Trademark Office*

[28] Object Management Group, "Semantics of a foundational subset for executable uml models specification, version 1.0," 2011. [Online]. Available: (http//www.omg.org/spec/FUML/)

[29] Ben Mena, S. Introduction aux méthodes multicritères d'aide à la décision, *Biotechnol. Agro. Soc. Env.* 4(2). 83-93, 2000.

[30] Olson, D. L. (2001). Comparison of three multicriteria methods to predict known outcomes. *European Journal of operational research*, 130(3), 576-587.

[31] Schärling, Alain, Décider sur plusieurs critères, Presses Polytechniques Romandes, 1985, 304 pages.

[32] Talbi, El-G., Méthodes d'optimisation avancées, *LIFL CNRS.* [Online]. Available: (http://www.lifl.fr/~talbi/Cours-optimisation.pdf)

[33] Zeleny, M., Multiple criteria decision making, *McGraw-Hill, Columbia University*, 1982, 563 pages.

[34] Lehoux, N., & Vallée, P. (2004). Analyse multicritère. *Ecole Polytechnique de Montréal.*

[35] Dehbi, Rachid, Talea Mohamed, and Abderrahim Tragha. "The generation approach of Multi-target learning management system." Advanced Science Letters 19.8 (2013): 2326-2330.

[36] Dehbi, R., Talea, M., & Tragha, A. (2012, July). LMSGENERATOR: Multi-target learning management system generator based on generative programming and component engineering. In Education and e-Learning Innovations (ICEELI), *IEEE, 2012 International Conference on* (pp. 1-6).

[37] DESIGNED, PARTICULAR PLATFORM USING AN EAV. MAPPING OF A PLATFORM SPECIFIC MODEL TO A PARTICULAR PLATFORM USING AN EAV DESIGNED PLATFORM MODEL. *Journal of Theoretical and Applied Information Technology*, 2015, 79.1.

[38] DEHBI, Rachid; TALEA, Mohamed; TRAGHA, Abderrahim. MDA-Based Transformation of LMS Business Components: The Contribution of XML Technologies and Model Transformations. *International Journal of Enterprise Information Systems (IJEIS)*, 2013, 9.4: 63-84.