MODEL DRIVEN ARCHITECTURE A REVIEW OF CURRENT LITERATURE

[1] AHMED MOHAMMED ELSAWI, [2]SHAMSUL SAHIBUDDIN, [3]ROSLINA IBRAHIM

[1] Ph.D. Candidate, Faculty of Computing, Universiti Teknologi Malaysia, Johor Bahru, Malaysia

[2] Professor, Dean, Advance Informatics School, Universiti Teknologi Malaysia, Kuala Lumpur, Malaysia

[3]Senior Lecturer, Advance Informatics School, Universiti Teknologi Malaysia, Kuala Lumpur, Malaysia

E-mail: [1]elsawi@gmail.com, [2]shamsul@utm.my, [3]iroslina.kl@utm.my

ABSTRACT

Among different Model Driven Engineering (MDE) approaches, the Object Management Group (OMG) adopted the Model Driven Architecture (MDA). The MDA approach is aiming to automate the software develop process by using models instead of conventional coding and bases on the separation of concern concept. In June 2014, the OMG released the second version of the MDA guide trying to realize the fundamental principles and to support the first MDA guide issued in 2003 with more detailed specifications. A gap of 11 years makes researchers from their own perspective and background come up with different interpretations of the MDA terms. The thing that generates a confusion of what is beyond and what is within the MDA scope. By strictly referring to the MDA standard (not to model-driven engineering in general), we provide in this work a review of the current MDA literature. We also shed the light on the MDA research directions, specifically on the automations of the MDA development process and its targeted platforms.

Keywords: *MDA, MDE, OMG, Model Driven Architecture, Literature Review*

1. INTRODUCTION

After the success in providing a technology independent infrastructure standardization (CORBA) [1], the OMG rapidly moved from its previous Object Management Architecture (OMA) vision [2, 3] by officially announced in 2001, the adoption of the Model-driven Architecture (MDA) approach. In order to motivate the MDA approach, the OMG supported the MDA by a standardized specifications for the Unified Modeling Language (UML) [4], the Meta-Object Facility (MOF) [5], XML Metadata Interchange (XMI) [6], and the Common Warehouse Metamodels (CWM) [7]. These specifications act as a core infrastructure of the MDA [8]. It is also served in realizing the separation of concern concept and increasing the degree of integration and interoperability between systems.

In 2003, the first guide to the MDA principles has been released by the OMG [9]. The guide provided a comprehensive definition for the MDA terms and considered as the technical reference for the MDA practitioners. A new version released by OMG in June 2014 that found much lighter in technical details comparing to the previous one [10]. Its sole purpose seems to make the business case for MDA. An eleven years of OMG silence, dragged the researchers according to their background to bring new terms and concepts from other MDE approaches under the MDA context. Consequently, new MDA adopters treat these terms and concepts as genuine MDA principles.

In this work, we are drawing boundaries on the MDA original concepts. We also positioned the MDA among other classical MDE and conventional software development approaches. This is beside a review of the current MDA literature. The work does not include a literature for other model driven approach.

The main objective of this work is that it provide a significant review for the original MDA principles specified by the OMG. Researches, or software developers who want to go for the MDA, this work will draw the track for their journey.

The next section of this work provides a review of the current MDA literature and the research directions with a focus on the MDA

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

development process and its targeted platforms. The review process presented in Section 3 and the discussion shown in section 4. Finally, we conclude with the findings, recommendations and future work in section 5

2. LITERATURE REVIEW

2.1 Model Driven Architecture

The vital objective of MDA is to derive value from models that support us compact the complexity and interdependence of complex systems. The separation of concern concept achieved by three architectural layers: At first layer, the computational independent model (CIM) captured the business and domain vocabulary that typically provided by the subject matter experts. This layer is bridging the existing gap between the domain experts and the system implementers. The second layer is the Platform Independent Model (PIM), which formalize the information in the CIM model independently from any technology platform. The Platform Specific Model (PSM) represent the third layer that focusing on the technical and platform implementation details. Despite its concern, the PSM is still considered too abstract to be executed on a computer platform. Consequently, the Platform Model (PM) emplaced to support the PSM and to act as a technical manual for the targeted platform. The PM implicitly exists in model transformation rules. The implicit employment of the PM during the transformation is limited the model transformation scope to address a single presumed platform [11]. The thing that create an uncertainty about the possibility of using this model mapping for other platforms different than the one for which it was designed. However, in [12] it has been realized explicitly to drive the transformation to a run-time environment.

Model's transformation or mapping is one of the major activities in MDA. Given the Platform Model (PM), it serves in transforming high-level models (PIM) to low level (PSM) models or vice versa. In addition, the transformation can be within the same level of abstraction (PIM-to-PIM) or (PSM-to-PSM). These scenarios of mapping classified as Model-to-Model (M2M) mapping or Model-to-Text (M2T). Both are under the MDA umbrella and supported by a good number of tools that furnished to address each scenario and mapping [13]. The transformation process automation between the MDA models is differentiated it from the other Model Driven approaches. This due to the fact that the approaches like Model Driven Engineering MDE, Model-Based Engineering and Model-Driven Development MDD, the models derived either as a communication means between the system's analysts and programmers, or to directly generate code out of models. Skipping the architectural layers of MDA and lacking the automation in the development process. Figure 1 Positioning the MDA among other model-driven based approach.

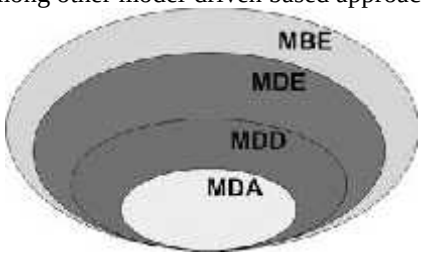


Figure 1: Positioning The MDA Among Other Approaches

The automation as well gives the MDA an upper hand in software productivity and cost reduction. Figure 2 (a) represent the classical software development process. Practically, any possible automation is taking place at the coding level downward. Consequently, changes in coding don't reflect the top levels, and updates on the top levels above the coding on the other hand means a considerable effort of recoding. This is because the text and diagrams in the above layers are commonly used for documentation and communication purposes [14]. The MDA process is shown in Figure 2 (b), present an automated closed loop of between levels. Where, changes in the abstract top levels propagated automatically to the lower ones, with minimal effort, time, and cost. On the other hand, changes in the lower abstract levels can be reflected in the upper ones, by reversing the transformation from lower level of abstractions code/artifacts to the upper layers with higher level of abstraction models [15].

2.2.2 Issues Facing MDA

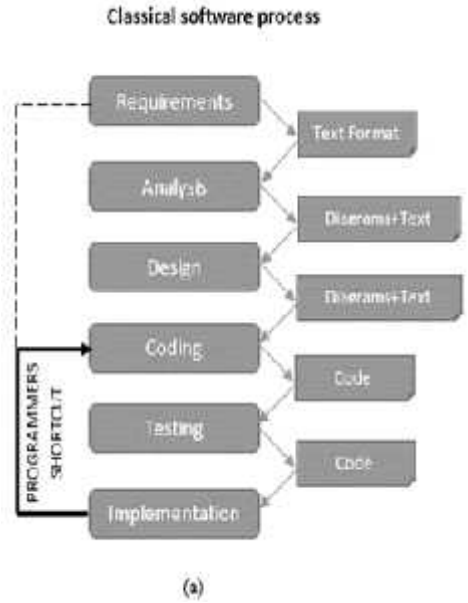
In the above section, boundaries drawn around what is MDA, and positioned the MDA among the other model-driven approaches. We also show its strength comparing to the classical software development and other model-driven approaches. Now we have a clear understanding of what is within the MDA scope and what is not. . In this part, we illustrate challenges and issues facing the MDA addressed by researchers.

Before the official adoption of the MDA by the OMG, the work in [16] present the dream and the advantage of the MDA, but they also

Journal of Theoretical and Applied Information Technology
10th September 2015. Vol.79. No.1

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

marked the technology diversity and dynamicity as the central issue that complicating system integration and interoperability. Which facing all the software development approaches and lifecycles including the MDA. The thing that create an urge to control the degree of abstraction of the MDA models in order to make these models executable at the targeted platform. Consequently, a chain of model transformation employed to address the platform environment. Meaning, a generation of new PSM models. Managing of these new models with its associated transformation rules and technique is not an easy task. It is not guarantee that enterprises stick on a single middleware like CORBA because of merging and new acquisitions of enterprises.



The model transformation is the key activity in MDA. There are different methodologies and tools addressing model transformation. A survey work by [17] focused on the transformation models and tools that support model to model transformation. While [18] is motivating Model to Text Transformations. As they provide an assessment of the MOFScript language, which submitted to the OMG as a proposed model to text transformation language. The existence of different transformation methodologies, which supported by some of the transformation tools in the market, make it a difficult for MDA practitioners to decide which tools are supporting which transformation methodology.

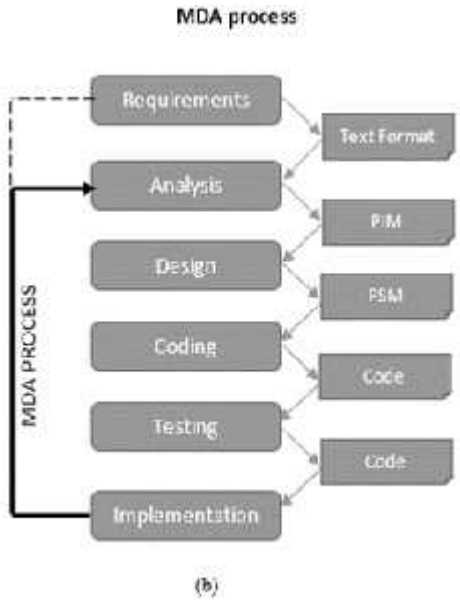


Figure 2: Classical Software Development Lifecycle Vs. MDA Lifecycle

On the other hand, the different level of abstraction in MDA lead to a chain of model transformation to go from high level of abstraction to lower one. The PM assumed implicitly in the model transformation. There is no guarantee that the model transformation can be used for other platforms than the one for which it was written. The PSM is targeting a particular platform. However, it is still considered too abstract to be executed on the platform. On the other hand, there is no platform similar to other. Even Java dragged to a technology diversity (J2SE, J2EE, J2ME, etc.). There is a need for the proper framework that capable to driven the transformation with minimum change and more details about the execution environment. These issues addressed by [19], as they used the ontology to define the platforms elements. However, the automation of handling significant ontology is impossible due to the number of classes and instances. This is beside the fact that the time consumed in a manual construction of ontologies is growing more complex upon the diversity and platform's data volume rapid increase. Subsequently, this is reflected in the number of ontologies required to cope with this technology volatility.

Another issue that the MDA is all about, is the automation of the development lifecycle and software productivity. Developers need automated techniques that are scalable, general, and extensible. However, the MDA process is a lack proper tools that can support the automation. The work in [20] proposes a framework to supports the

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

MDA automation directions. Beside their presentation to the platform component, they provide a benchmark to the tools that support the MDA automation vision.

2.2.3 MDA in Action

In view of the above issues, new directions motivate for bypassing some of the MDA layers or adding new ones. The work in [19] is motivating for eliminating the PSM and address the targeted platform directly from the PIM. It is similar to the agile methodology adopted by[20]. Other agilest like [21] are chasing the dream of having an executable model without any consideration of the MDA principles and by going directly for the UML 2.0 profiles, which provide a degree of flexibility in expressing and address platform's elements. The Agile MDA is an existing concept that becomes more attractive for software firms as model's role shifted from documentation to actual execution and implementation. Despite its promises, the MDA described as a heavyweight process that can lead to delivering the wrong system late at great expense. The agile methodology motivates for bringing the concept of that the code and model are operationally the same. Where an executable model can iteratively be constructed, run, test and modified in short cycles [21].

On the other hand, we found approaches like ArchMDE [22] is trying to create a new standard that allow a creation of software architecture from the analysis model. As they are proposing to add a new layer (AIM) in the MDA concept. Other researchers are suggesting of employing the CASE tools as a different path that mixed between the MDD and MDA [23]. The classic CASE Tools designed based on a traditional database concept. Where models stored in a retrievable repository. Our work in [23, 24] allow us to have the flexibility of representing models in a textual format that can be transformed into XML format. However, we remained within the MDA boundaries. The Textual format gives more control in the model constraints and dependency management, and it is supported by the MDA specifications.

Similarly, the work in [25] provide an End to end transformation framework from PIM to PSM within the MDA scope to support the software product line. Also, the work in [26-28] demonstrated a high degree of commitment to the MDA principles.

3. REVIEW PROCESS

The review process followed in this literature based on scanning the OMG MDA database[1], and its specification's section[2]. MDA, and other model-driven approaches in other scholarly databases like Google scholar, Springers, IEEE Explorer and ACM were scanned as well. We go through an operational definition process to define what within the MDA scope by referring to [9, 10] as a ground base to compare the MDA approach with other model-driven approaches based on the following 4 criteria:

- The adoption of the separation of concern concept.
- The MDA architectural layers conformation.
- Transformation methodology.
- Handling Platform Diversity vs. the automation of the development lifecycle.

Figure 3, representing the high level of the MDA concept according to the OMG standards. The concern divided to business domain that include the CIM and PIM. While the PSM and PM are, platform and technology focused. The transformation area split to transformation rules and transformation tools. Both should support the control of model's abstraction degree.

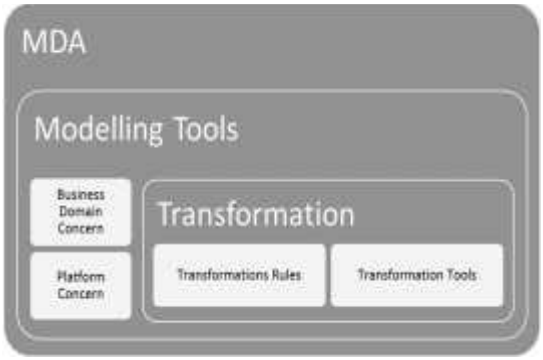


Figure 3 An MDA High Level Architecture

Researches in this review that formulate a hypothesis of modifying the current MDA architectural layes addressed by reviewing a relevant type theme and researches in model based software development process area. The evaluation criteria for the reviewed literature based on the high-level architecture in Figure 3. In addition to the two MDA guides released by the OMG.

[1] http://www.omg.org/mda/presentations.htm
[2] http://www.omg.org/mda/specs.htm

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

4. DISCUSSION

While the MDA aiming to design and build software product that can run in multiple platforms with minor engineering by adopting models, there is other model centric approaches like MDE and MDD. On the highest possible level, these approaches are aiming to enhancing the software quality, productivity, interoperability, cost and time to market. However, in practice, both approaches operate on different levels to achieve the same goals. The MDA adopting models in a higher separately concerned abstraction levels to produce software, avoiding hard coding in an end to end automation. While the other model driven approaches not strictly focused on the process automation or the separation of concern. Thought, the models at some point used for documentation or communication purpose.

The absence of the OMG guidance in realizing the MDA concepts dragged the researchers to come up with their own initiatives from other model driven approaches based on their own prospective and background. Consequently, a confusion state dominate the research outcome and the MDA basic principles compromised.

The Agile MDA is a quite interesting area that we discussed in this review. Especially in the part of the model merging. The PSM can merged with the PM as both are models and can represented either textually or graphically. Elimination of the PSM part or adding new layer to the MDA process are lacking a tool to support. The MDA standard tools are not equipped to handle such initiatives. QVT focusing on model-to-model transformation. To have executable code we need to adopt MOF model to text. Which consequently, we will be limited to a few number of language such as EMF or ATL. or ATL.

5. OPEN RESEARCH ISSUES IDENTIFIED

Different current directions in research are in this review, which divided to two: The first direction focusing on the MDA infrastructure and the tools that support its principles. Researchers taking this direction have a limited space to work in, if they want to strictly stick on the MDA principles and get the OMG approval for their approaches or tools. This is because the productivity of the OMG in providing the MDA adopters with enhanced or new specification is very poor. Consequently, different tools appeared that achieve many of the MDA basics. However, it is not necessarily following the OMG specifications. The Model-to-Text is a good example for this situation. The standard is there but the tools and researches that following these specifications is very limited. Consequently, new MDA joiner are desperate for tools that provide a decent level of documentation and support.

The second research direction focusing on the MDA applications. This direction grow very fast and reach a good degree of maturity. Different MDA applications and researches on the area of cloud computing, software testing, and embedded system outcome a very interesting results and success stories.

6. CONCLUSION AND FUTURE WORK

Computer platforms are combining a set of features and component that allow to formally controlling different functions and contexts (Hardware/software activities) that normally limited to a particular stream of technology. Having a capability of producing software out of models to support multiple platforms for different technologies in different levels of abstraction is dramatically increasing the quality and the productivity of the software development process. The MDA principles can contribute positively in software quality and productivity in general, and in particular, how it is different from other model centric approaches. The importance of this work that it can be used to understand the current position of the MDA and how to follow its principles. In addition, for those researchers who want to take the other approaches directions can use this review to maintain and position their work.

The issues that facing the MDA progress has been reviewed and presented in this work. Interestingly these issues is facing all the other model driven approaches in away or other.

In the near future, we will provide more work on the MDA automation area specifically, on the transformation from platform specific model to computer-targeted platform adopting model to text OMG standard.

REFRENCES:

[1] Corba, O., *IIOP 2.3 specification*, 1998, 1998.
[2] Siegel, J., *OMG Overview: CORBA and the OMA in Enterprise Computing*. Communications of the ACM, 1998. **41**(10): p. 37-43.
[3] Soley, R.M. and W. Andreas, *Object Management Architecture Guide: Revision 2.0*. 1995: Wiley.

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

[4] formal/01-09-67, O.D., *OMG: Unified Modeling Language v1.4. OMG Document: formal/01-09-67, Sept.2001.*, 2001.
[5] OMG Document: formal/01-11-02, *OMG: Meta Object Facility (MOF) v1.3.1*, in *OMG Document: formal*2001.
[6] OMG, X., *Metadata Interchange (XMI) Specification*. URL: http://www. omg. org/docs/formal/05-05-01. pdf (accessed October 10, 2005), 2000.
[7] Chang, D.T. *Common Warehouse Metamodel (CWM), UML and XML*. in *Meta Data Conference (March 19-23, 2000)*. 2000.
[8] Poole, J.D. *Model-driven architecture: Vision, standards and emerging technologies*. in *Workshop on Metamodeling and Adaptive Object Models, ECOOP*. 2001.
[9] OMG, *OMG: MDA Guide Version1.0.1*. 2003. **Version 1.0.1**.
[10] OMG, *MDA Guide revision 2.0*. 2014.
[11] Soares, I.W., et al., *Modeling of embedded software on MDA platform models.* Journal of Computer Science & Technology, 2012. **12**.
[12] Almeida, J.P., et al. *On the notion of abstract platform in MDA development*. in *Enterprise Distributed Object Computing Conference, 2004. EDOC 2004. Proceedings. Eighth IEEE International*. 2004. IEEE.
[13] Mellor, S.J., *MDA distilled: principles of model-driven architecture*. 2004: Addison-Wesley Professional.
[14] Dobing, B. and J. Parsons, *Dimensions of UML diagram use: a survey of practitioners.* Journal of Database Management (JDM), 2008. **19**(1): p. 1-18.
[15] Czarnecki, K. and S. Helsen. *Classification of model transformation approaches*. in *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*. 2003.
[16] Soley, R., *Model driven architecture.* OMG white paper, 2000. **308**: p. 308.
[17] Gerber, A., et al., *Transformation: The missing link of MDA*, in *Graph Transformation*. 2002, Springer. p. 90-105.
[18] Oldevik, J., et al. *Toward standardised model to text transformations*. in *Model Driven Architecture–Foundations and Applications*. 2005. Springer.
[19] Wagelaar, D. and R. Van Der Straeten, *Platform ontologies for the model-driven architecture.* European Journal of Information Systems, 2007. **16**(4): p. 362-373.
[20] Jackson, E.K., et al. *Components, platforms and possibilities: towards generic automation for MDA*. in *Proceedings of the tenth ACM international conference on Embedded software*. 2010. ACM.
[21] Mellor, S.J., *Agile mda.* MDA Journal, 2004.
[22] Elleuch, N., A. Khalfallah, and S. Ben Ahmed. *Software Architecture in Model Driven Architecture*. in *Computational Intelligence and Intelligent Informatics, 2007. ISCIII'07. International Symposium on*. 2007. IEEE.
[23] ELSAWI, A.M., S. SAHIBUDDIN, and A. ABDELHADI, *PROPOSE AN INTEGRATION BETWEEN UML STATIC AND DYNAMIC MODELS USING ENTITY-ATTRIBUTEVALUE UNDER THE MDA CONTEXT.* Journal of Theoretical & Applied Information Technology, 2014. **68**(1).
[24] ELSAWI, A.M., S. SAHIBULDIN, and A. ABDELHADI, *INTRODUCING THE OPEN SOURCE METAMODEL CONCEPT.* Journal of Theoretical & Applied Information Technology, 2013. **57**(3).
[25] Hamed, A. and R.M. Colomb, *End to End Development Engineering*. JSEA, 2011. **4**(4): p. 195-216.
[26] Mukhtar, M.A.O., M.F.B. Hassan, and J. Bin Jaafar. *Optimizing method to provide model transformation of model-driven architecture as web-based services*. in *Computer & Information Science (ICCIS), 2012 International Conference on*. 2012. IEEE.
[27] Mukhtar, M.A.O., A. Azween, and A.G. Downe, *A Proposed Compiler to Integrate Model Driven Architecture with Web Services–Road Map.* International Journal of Computer Applications, 2011. **15**(7): p. 1-7.
[28] Mukhtar, M.A.O. and A.B. Abdullah, *Mapping of Behavior Model using Model-Driven Architecture.* International Journal of Computer Applications, 2011. **13**(8).