A Framework for Model Driven Transformation Engineering towards Software Architecture and Performance

Ahmad Mateen
University of Agriculture
Faisalabad, 38000 Pakistan

Alia Tabassum
University of Agriculture
Faisalabad, 38000 Pakistan

ABSTRACT
This paper presents a different experience and application of defining and implementing an Agile Development Process (ADP) using Model Driven Architecture (MDA) series. This process and its implementation inherit the merits of both methodologies. The research demonstrates that combining MDA practices with ADP can significantly moderate software development cycle time and increase productivity and quality. This approach offers several other advantages including Platform-independent models which are used to promote the system endurance and flexibility in deployment. Executable models are assembled to increase the level of consideration, quality and efficiency. Change of models that are utilized to implement and allow the advanced run-time execution through programmed improvements that is not practical with transcribed code. The principles of modeling describe that the transformation should also explicitly modeled because it is at the very core of Model Driven Engineering. Transformations allow synchronization, analyses of execution, optimization, code synthesizing, composition and developing models.

Keywords
Agile Development Process (ADP), Computation Independent Model (CIM), Model Driven Architecture (MDA), Platform Independent Model (PIM), Platform Specific Model (PSM).

1. INTRODUCTION
Agile process methodologies for software development take an innovative, lightweight approach features in designing and producing applications. These practices solve the common problems of developing software by encouraging and facilitated the direct user involvement, small and frequent release, and rapid iterations by evolving requirements of client or customer in the development process [1].

The numerous agile software development approaches have advanced following their starting and even requested high degree to enhance the nature of the product item. Some of the well-known agile development strategies are Extreme Programming (XP), Scrum, Crystal Methods, Feature Driven Development (FDD) and Test Driven Development (TDD). These approaches are not the same as conventional programming procedures and help the associations to meet the difficulties of this period. [7]

The Model Driven Architecture (MDA) is a structure for programming improvement characterized by the Object Management Group (OMG) [6]. Inside MDA the product advancement every procedure is driven by the activity of modeling the software system which provides the benefits of

Productivity, *Portability* and *Interoperability*. There exist a few MDA-based [3] and additionally agile [8] approaches in the literature; however a MDA and agile development are not a solid programming improvement procedure. Accordingly, with regards to develop software with an agile development process using model driven architecture [4] it is feasible either to introduce agility into current MDA-based strategies through agile practices and rules or to incorporate MDA standards and development tools into existing agile programming, which are both evident samples of extension based system designing methodology [3].

2. BACKGROUND
2.1 Agile Process Development
Requirements constantly changes according to the business need. In this way, programming architectures ought to have the capacity to oversee business forms and have the ability to meet the future modifications and business necessities. The field of programming improvement reports precisely the difficulties of an eccentric, unsettled business and innovation environment. In this manner, the inquiry is how to better switch architectural modifications for achieving high quality [9].

Principles of Agile Philosophy [3]

- Higher customer satisfaction
- Facilitation of changes in requirement
- Frequent delivery of work in a shorter time.
- Collaboration of client and developer
- Self-organized expert team
- Direct conversation of team members
- Main extent of progress is working software.
- Agile processes support maintainable progress.
- Constant consideration to nominal excellence and good design of the product
- The best architectures, requirements and designs
- At regular intervals the main focus on how to become more effective in productivity and quality

2.2 Model driven Architecture
One of the fundamental points of the MDA is to segregated configuration from construction modeling. As the ideas and innovations used to acknowledge outlines and architectures have changed at their own pace, blending them permits framework designers to look over the best and most fitting in both areas. The design reports the useful prerequisites while architecture gives the foundation through which non-

functional necessities like versatility, performance and reliability are figured out [6].

From the conventional life cycle, the MDA advancement life cycle does not look altogether different. The items are formal models, similar to a few models can be comprehended by computers so one of the significant contrasts is the way of the work that is made during the improvement process [2].

3. RESEARCH PROBLEM

This study states an important issue in software engineering that is a gap in IT and Business. To improve the development process of software systems; there should be a framework to produce new competencies that can help to reduce the gap effectively.

4. ANTICIPATED CONTRIBUTION

For project development, a framework that relates MDA to ADP will be presented as a complete system in this paper. To test this framework a number of case studies will be used in the form of small software projects.

5. MODEL TRANSFORMATION

A typical methodology towards making adaptable, dynamic business forms and light-footed application is the service oriented computing style a dispersed frameworks is key to make utilization of an adaptable and versatile platform that can react to new necessities in a proficient way. In this manner, the use of suitable architectural styles for the outline of programming frameworks is a task [9].

Conversely, MDA changes are constantly executed by tools whereas customarily the changes from model to model, or from model to code, are done principally by hand. There are numerous tools that have possessed the capacity to change a PSM to code; there is something new in MDA is that the change from PIM to PSM is mechanized too [2].

It can incorporate any modeling language that is utilized with domain specialists with MDE by aiding from mechanized model verification and transformation. MDA does not maintain a strategic distance from transformative strategies with an iterative and incremental advancement process. These can utilize the MDA tool to create and continue iteratively and incrementally without issues, especially without MDA-related ones [10].

All MDA modeling language are formally determined in Model Object Facility (MOF) which is a stone strong formation. UML utilized for model representation are likewise standardized, similar to the specification language for model changes. UML was intentionally kept brief, without area specifics rather it offers lightweight extensibility, allowing the production of UML profiles for meta models formally determined in the MOF. It can make profiles naturally utilizing MDA [5].

The idea of MDE is to constrain the adaptability for effortlessness and profitability. Subsequently it is unrealistic to make each desired thing however this is never an issue unless it utilize a traditional methodology; in fact, components outside the tools were the issue. It did lead me to the assumption that if MDD device achieves a sure level then improvement is not the troublesome part any longer. When there is a model then from just with a single tick send to make the last application. However the testing part is to turn thoughts and/or business issues into such a model. Imperative components in this procedure are requirement gathering and project management.

6. BASIC FRAMEWORK

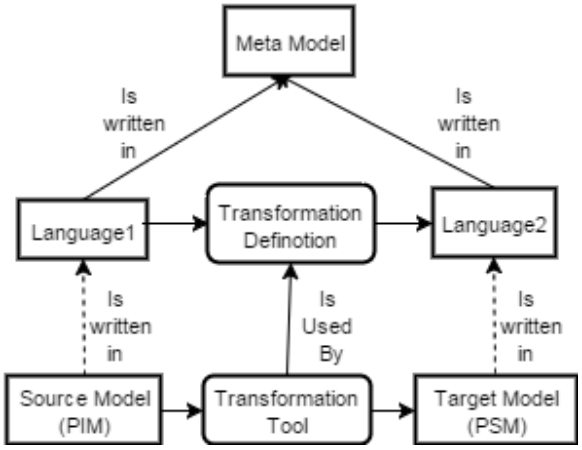

Fig 1 Overview of the basic MDA framework

Fig.1 depicts major elements that participate in the MDA framework: models, PIMs, PSMs, languages, transformation definitions, and tools that perform transformations. These components fit together in the essential MDA structure. it summarize the elements and their role below:

- A *model* is a depiction of a framework written in all around characterized language.
  - *Platform Independent Model* which defines framework with no learning of the last usage stage
  - *Platform Specific Model* which defines framework with full information of the last usage stage.
- A *transformation definition* defines how a model in a source dialect can be changed into a model in target dialect.
- A *transformation tool* performs a change for a particular source model as indicated by a change definition [2].

The PSM and PIM are the most key roots from the designer's perspective. A designer puts his attention on adding to a PIM, which make the product framework at high level of abstraction. In the next stage he chooses one or more mechanisms that can perform the change on the PIM that has been developed according to certain requirement.

7. PROPOSED APPROACH

Proposed experiment case is to relate MDA approach with Agile Development principles which are used in the process of creation of PIM. With the agile methodology it start from a simple model and make the model reusable.

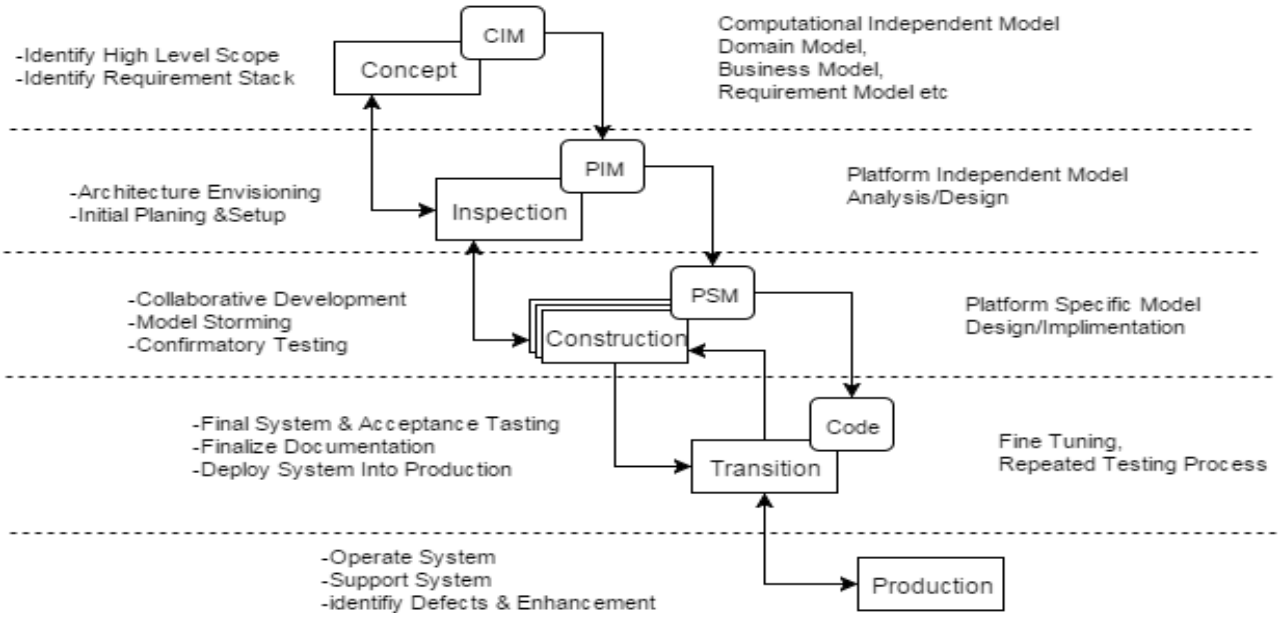

Fig 2 Overview of the framework

For this purpose there is a need to create a generic PIM which can be adapted to the specific software system when needed. The following components need to be developed:

- Use Cases
- State Model (state diagrams)
- Class Diagram: Entity and Relationship (their attributes and integrity constraints)
- System Behavior Model (operations)

In the development process, just before a complete reusable model for different project, it has to include different kind of models and approaches to enhance the productivity. Depending on particular project there can be different background knowledge like domain model. To support consistency between all models, principles of agile modeling and MDA should be applied.

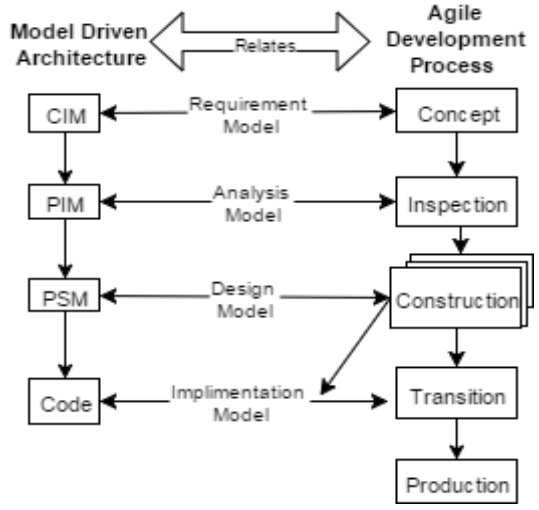

Fig 3 MDA& ADP relation

The Fig. 3 identifies some *Executable Models* as following:

1. *Requirement Model*: its aim to organize and monitor the project requirements according to CIM and Concept phase.
2. *Analysis Model*: its aim to identify modeling and transformation needs according to PIM and Inspection phase.
3. *Design Model*: its aims to obtain the modeling and transformation specifications according to PSM and Construction phase.
4. *Implementation Model*: its aim to tool provision and metadata administration facilities prepared to utilize Code and Transition phase.
5. *Project execution*: its aim to produce the vital programming artifacts and the final items.

Fig.3 portrays how the arrangement has been organized in diverse stages. These stages are helpful to comprehend and to

represent the conditions between the workouts. The periods of this philosophy identify with the accessible and required

aptitude and accordingly these stages can be specifically connected with the dividing of the ADP and MDA. 1st and 2nd stage mainly performed by knowledge facilitators, 3rd and 4th stages are essentially performed by information

constructers while 5th stage is mostly performed by learning clients. It additionally demonstrates that numerous conditions are recognized between the development periods of this strategy, which implies that these stages ought to be performed

iteratively and incrementally so the criticism from the execution exercises to the arrangement exercises and the other way around ought to be performed in a powerful way. This accessibility could be known as Transformation of model in which model-to-model changes, code generation systems and very much characterized traceability methodologies are essential. Continuous integration and testing are two vital issues to be followed in this framework.

8. RESULTS

A number of small projects related to academics are used as case studies that are shown in Table 1. The table demonstrates

that in CIM the use case diagrams are mostly used. Whereas all UML models aside from component and deployment models are used in PIM while in PSM there are class and component diagrams. Fig. 4 classifies the executable molding diagrams in project case studies. From the accepted agile principles and perspective, various projects articulated the standard setup of code and provided meaningful remarks. Which will also shows It gets to be conceivable to straightforwardly move from model to code with the emergence of MDA tools, because changing a model means changing the software thus the MDA approach can support ADP. Just like the source code, this spreads models from being a part of documentation simply turning out software design. Without a doubt a model is just worth building in the event that it specifically accomplishes the last objective of building a working framework

Table 1. Modeling Diagrams in Project Case Studies

Project Title	Process Model	CIM	PIM	PSM
Student Records System	Scrum	Use Case	Activity Diagram	Class Diagram, Sequence Diagram
Career Path	XP	Use Case, DFD	Activity Diagram, Sequence Diagram, DFD	Class Diagram
Hospital Information Management System	Scrum	Use Case, DFD	Sequence Diagram, Class Diagram, DFD	Component Diagram
Library Management System	XP	Use Case	Sequence Diagram, Class Diagram,	Component Diagram
Pizza Order Management System	XP	Use Case	Class Diagram, Activity Diagram	Component Diagram

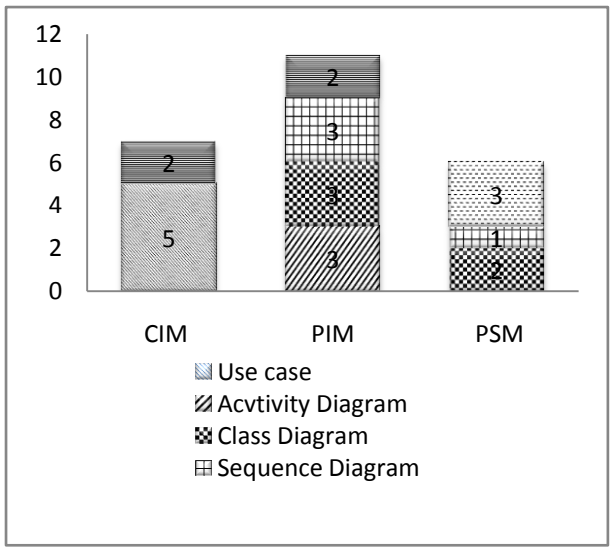

Fig 4 Modeling Diagrams in Project Case Studies

9. CONCLUSION

This paper provides the idea of combining MDA approach with ADP methodologies. It would ponder that the advantages of MDA are intensely linked with the utilization of an agile methodology. By identifying the well formalized knowledge it would be prove that MDA and agile are content pair. Because of the utilization of high state models MDA empowers close shared effort with the customer and provide short repetitions because much of the development process is programmed. Future work around there can prompt another framework which could be connected different methodologies of agile development process..

10. REFERENCES

[1] Ahmed A., Ahmad S., Ehsan N., Mirza, E.and Sarwar S. Z., 2010. "Agile Software Development: Impact on Productivity and Quality," in Proc. ICMIT IEEE Software Int. Conf., pp. 287 – 291.

[2] Kleppe A., Warmer J., Bast W., April, 2003. The Model Driven Architecture: Practice and Promise, Addison Wesley, pp. 45-68.

[3] Beck K., Petal, 2001. "Principles behind the Agile Manifesto", Agile Alliance, Available: http:// agilemanifesto.org/principles.html[last visited 28-7-15]

[4] Y. Zhang and S. Patel, "Agile Model-Driven Development in Practice," *IEEE Software*,vol. 28, no. 2, pp. 84-91, Apr. 2011.

[5] D. D. Ruscio, L. Iovino, and A. Pierantonio, "Coupled Evolution in Model-Driven Engineering," *IEEE Software*, vol. 29, no.6, pp. 78-84, Nov2012.

[6] G. Bergmann, I. Ráth, J. Varró, and D. Varró. "Change-driven model transformations,"*Software and Systems Modeling*,vol. 11, no.3, pp. 431-461, July 2012.

[7] "Introduction to Agile Development" Available: www.serena.com/docs/repository/solutions/intro-to agiledevel.pdf [last visited 3-8-15]

[8] T. Kühne, G.Mezei, E.Syriani, H.Vangheluwe, and M.Wimmer,"Explicit transformation modeling. Models in Software Engineering," in *Lecture Notes in Computer Science: Models in Software Engineering*, vol. 6002, no. 10, pp. 240–255,Oct. 2009.

[9] R. Mordinyi, E. K¨uhn, and A. Schatten., "Towards an Architectural Framework for Agile Software Development," in *Proc. ECBS IEEE Software Int. Conf., 2010*, pp. 276-280.

[10] Woodside M., Franks G., and Petriu D. C., "The Future of Software Performance Engineering," In Proc. 2007IEEE Int. Conf. Software Engineering, vol. 32, no.10, pp. 171-187.