ISSN-L: 2223-9553, ISSN: 2223-9944
Vol. 4 No. 1 January 2013
Academic Research International

ADAPTABLE MODEL-DRIVEN ENGINEERING FOR FORMAL METHODS INTEGRATION WITH AGILE TECHNIQUES FOR DESIGN OF SOFTWARE SYSTEMS

Kebande Rigworo Victor

Department of Computer Science,
Egerton University, Egerton, Nakuru,
KENYA.

vickkebande@yahoo.com, vkebande@egerton.ac.ke

ABSTRACT

An MDE for integrating Formal methods and Agile software development technology is an analogy for combining the complex computer based formality with systematic agile process.MDE principles are adopted which vividly shows how different tools can be called to action. This action aims at extracting a mathematically rigorous representation of a composite scheme that enables one to prove how the assimilation of the two orthogonal animals, Agile and Formal methodology works. Approaches and mechanisms are followed thereafter different methodologies and factors about the acknowledged synergy at various stages in its development.

Keywords: Formal, Methods, Agile Model Driven Engineering

INTRODUCTION

Software models mainly consist of relic in the software engineering process. This process has to undergo many stages for effectiveness, and the evolution of a software system can be interpreted in many ways, ranging from manual bug-fixing to self-adapting software that can alter its behavior depending on changes in the environment. Agile methods will try to weave Douglas et al., into formal methods through a model structure so, to avoid the act of relying on a distinct piece of information in order to get executable software; I suggest a new paradigm that is to be followed. In this view the MDE (Model Driven Engineering) is what I adopt as a paradigm to help realize a way through which formal methods software development tools can rely on Agile methodologies, Schmidt, *Slaby,Model driven Engineering. 2006.* MDE promotes the use of models as higher-level artifacts, the separation of concerns and generative approaches. The Adaptable MDE framework gives a promising approach to add value in analyzing the development of the software systems within the program life cycle and adaptability will aim at moving away from the traditional approach by the influence of formal methods and agile methodologies. Agility focuses on best practices and methodologies for programming of software systems and their integration *Mahé , Benoît , Cadavid. Crossing MDE 2009,* within a development process of software using formal methods can increase the quality of software designed. Although what exists shows that fewer formal methods can align to the agile process. Contemporary development practices shows that agile development methods would change rapidly given that the requirements proposed at the start must meet the customer's requirements. From this they follow an approach with high customer interaction during the entire development cycle. Consequently they are more open to changes if errors occurred in the specification of the original requirements. Agile software development even thrives on this by eliciting and refining requirements. The software hence grows incrementally and is constantly being refined. It is used actively to communicate the developer's understanding to the customer.

ISSN-L: 2223-9553, ISSN: 2223-9944
Vol. 4 No. 1 January 2013

OVERVIEW OF FORMAL METHODS

Formal methods approach relies on the use of meta-mathematical approach to solve software engineering problems at different levels of specification and design, correctness, they are considered complex but more reliable and correct computer based systems, (Eleftherakis, & Cowling, 2003), basically they are mathematical practices, often supported by tools, for emergent software's and hardware systems. Mathematical techniques in adaptable MDE enables the user to analyze and verify these models at any part of the program life-cycle .The phases being analyzed include the following; requirements engineering, specification, architecture, design, implementation, testing, maintenance. They (formal methods) solve the problems of reaction to the face up in computer-based systems, and the fault that arise as a result. They are a method used to model and analyze complex computer-based systems as mathematical entities.

By producing a mathematically rigorous model of a complex system, it enables developers to verify or prove false claims about the acknowledged system at various stages in its development. Formal methods can be applied to models produced at any stage of a system's development.

OVERVIEW OF AGILE TECHNIQUES

Agile software development is a group of software development methodologies based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams as laid out in the 12 principles behind the Agile Manifesto, *wwwagilemanifesto.org*. The Methodologies behind Agile development is that, Agile have been developed to address the issue of bringing up high quality software under the regularly and fast changing requirements and business environment. Agile development processes are portrayed by wide-ranging coding practice, rigorous communication between key stakeholders, small and flexible teams and fast iterative cycles. Formal correlations between Agile will produce optimized collaboration with Agile, (Overdick, Puhlmann, Weske, 2005). With the introduction of extreme programming the biggest challenge is noted at the change in software requirements, but agile processes acknowledge the certainty of change against the hunt for whole, strict specifications.

Extreme programming simple rules helps keep their design simple and clean as agile software development solutions are targeted at enhancing work at project level.

AGILE TECHNIQUES INCLUDE

1. Adaptative Software Development (ASD)
2. Feature Driven Development (FDD)
3. Dynamic Software Development Method (DSDM)
4. Rapid Application Development (RAD)
5. Scrum
6. Xtreme Programming (XP)
7. Rational Unify Process (RUP)

Adaptative Software Development (ASD)

Is an Agile software development, it grew from the rapid application development

Feature Driven Development (FDD)

Is an iterative and incremental software development process. It is one of a number of Agile methods for developing software

Dynamic Software Development Method (DSDM)

Dynamic System Development Method is an approach to system development, which develops the system dynamically. This methodology is independent of tools, in that it can be used with both structured analysis and design approach or object-oriented approach.

Rapid Application Development (RAD)

RAD is a process in software development that builds usable systems quickly (less than six months)

Scrum

Scrum is an agile framework for completing complex projects. It is an iterative, incremental framework for project management often seen in agile software development.

Xtreme Programming (XP)

Xtreme programming is one of Agile process. It is a discipline of software development based on values of simplicity, communication, feedback, and courage. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation.

Rational Unify Process (RUP)

It is a software engineering process, aimed at guiding software development organizations in their endeavors. Agile tools must be implemented after observing the underlying system, properties and how it will be represented at large. These tools are able to change the system structure without changing its behavior at all cost.

MODEL DRIVEN ENGINEERING (MDE)

I first introduce some basic terminology of MDE**.** A model of a system is a depiction or plan of that system and its setting for some convinced purpose. Also they are sets of objects which types are defined in metamodels. They can also be seen as graphs of objects interconnected by relationships. i.e abstract representations of the knowledge and activities that govern a particular application domain MDE will always increase productivity in development *Mahé , Benoît , Cadavid.Crossing MDE 2009,*A platform is a set of subsystems and technology that provide a rational set of functionality through interfaces and precise usage prototype, which any application sustained by that platform can use without worry for the details of how the functionality provided by the platform is implemented. It is model-driven as it provides a way of using models to express the course of understanding, design, construction, deployment, operation, maintenance and modification. MDE aims to improve efficiency of developers by using the modeling languages and simplifying the design process.

MDE PRINCIPLES IN AGILITY

From the MDE principles the expectation should be agility and higher level of abstraction, formal verification and lithe methods of software engineering.

The principles to be followed in MDE should achieve the following.

1. High-quality design
2. Operational software
3. Simplicity
4. Software should be dynamic
5. Effective communication between the Agile process and formal design

ISSN-L: 2223-9553, ISSN: 2223-9944
Vol. 4 No. 1 January 2013

The MDE splits itself into two abstract methods for effective communication in Agility

1. Meta-modeling(agility)
2. Modeling (agility)

Meta-modeling represents the actual definition or modeling of languages and their respective modeling tools. Modeling represents using the available tools to engineer the final procedure of these applications.

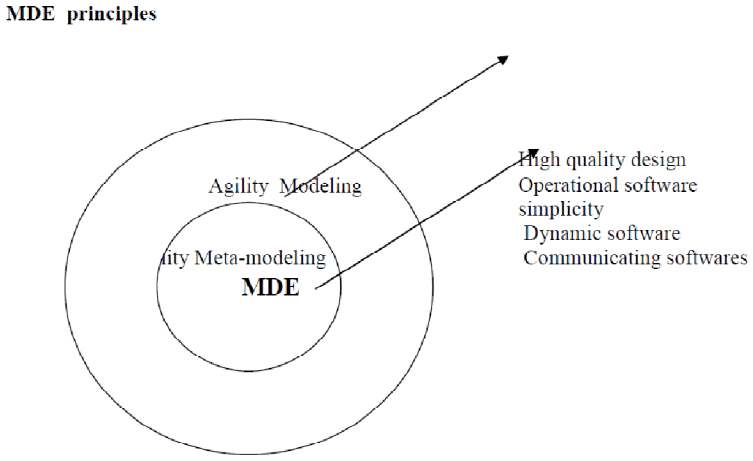

Figure.1 MDE framework for Agility modeling and Meta-Modeling

Agility is introduced in meta-modeling and modeling to openly show all the conceptual work and the whole implementation of the final applications being modeled. Using this approach of MDE different principles can be brought together because models are introduced initially for the sole aim of analysis drawing to help capture the requirements, but nowadays model execution is the key issue of modeling activities. If we rely on verification and validation w.r.t MDE, we can conclude that it aims at telling us whether the product we are building is right, at higher levels of this development if models are embedded then we can specify the system that is being built.

MDE Factor In Integrating Formal Methods And Agile Development Process.

Relying in the MDE to integrate the formal methods into Agile process would indicate that models will confirm and emphasize the techniques which can either be some management practices in getting the codes executed easily as the issues have been brought into focus in IEEE Computer in formal methods are predominantly effective early in improvement at the requirements and specification levels(Paul et al., 2009). Though Formal Methods and Agile methods are different (Black et al., 2010), Variations of using formal methods theories are a way to verify and validate specifications to ensure that there is some correctness of the software. In the other word is formal methods is a strict mathematical definition of the effect of the required maneuver and relatively is an expensive process. For the reason that we can start our model from a simple module and as long as it's the module that is going into iterations of the model. The model also is going into iterations until we get a complete module confirms to a concrete model. The development process is always designed to have long sprints for development and it has to reflect the same for formal methods.

These practices are the implementations of the different sets of agile principles of which their aim will be to offer guidance when performing some technical management. Due to the

prevailing software crisis integration of formal methods and Agile methods can hold the key to this as they may come up with a solution to faster development and quick error detection (Gruner, 2009/2010). Traditional software development methodologies had realistic big portion of time which was entirely spent at the beginning of the project; this was basically for requirement gathering and specifications definition. These methodologies were mainly developed in order they could help to apply formal methods theories. If you follow agile software development it was a challenge to combine our software development with formal methods. The approaches of formal methods and agile techniques are both different when it comes to software design, because formal methods are regarded as expensive way of coming up with software, if the techniques are merged with agile methods which are solely concerned with development and delivery. The Variations of applying formal methods studied and modeling techniques is expected to succeed among the other techniques. In this experience the nature of modeling is the key to make this combination between agile development and formal methods possible. The correctness of formal methods must depend on the channel of productivity of Agile methods and the different methods used in this development with reference to extreme programming, if we are going to rely on this two methods for software design, then for Agile different tools must be adopted to spearhead the development process. Example; the use of CASE tools and UML for Agile development can be adopted.

CASE Tools And UML For Agile Development

Agile development (AM) being a practice based methodology, CASE tools often promote syntax over communication between developers in other words, the model might look good but doesn't necessarily work. So the generated code has to be often too simple or cluttered with extraneous information required by the tool. The Agile development requires high level requirements which can be represented using the notations of UML activity diagrams, which will indicate the tasks of the developers and project stakeholders. The Unified Method Language (UML) shows the strong link between modelisation form and prototype development.

FORMAL METHODS SURVEY.

Formal methods are basically used in software design. As real engineers use mathematics, software engineers use formal methods to design software's since complex software systems requires organization, different methods and techniques should be followed when self organizing this formal methods, this includes ;

- Formal specification
- Formal verification
- Model checking
- Theorem proving

Formal specification will give a highlight of the general system properties; including the behavior, performance and characteristics. The trends to be followed must carry all the aspects of the system being described. This includes the domains contained in the state and transitions and behaviors.

Verification brings out the correctness involved with regard to what is being exposed at formal specification. Here you have to prove the truth behind the description outlined in the specifications. Model checking involves performing an immediate search to the verified problem theorem proving is the developing of proofs logic and ensuring it conforms to the system and its desired properties. At the implementation level formal methods are used for verifying the code and at this level every program must show some correctness as that's what

ISSN-L: 2223-9553, ISSN: 2223-9944
Vol. 4 No. 1 January 2013

natural formal methods do. If you were to work on 1000LOC, correctness of this LOC must be felt and realized at the implementation level. In the survey it is evident that formal methods have replaced traditional engineering design methods. In the future the expectation are that emphasis will be placed on integrated formal development support environments which are intended to support most formal stages.

EVALUATION PLANS

Committed model versioning approaches are adopted as a way of evaluating how formal agility can be achieved given that the production velocity of this Agile methods must be systematic, which openly operate on the models and not on the textual representation. However, this evaluation approaches suffers from three major deficiencies. First, they each support only one modeling language or, if they are generic, they do not consider important specifics of a modeling language. Second, they do not allow the specification of composite operation such as refactoring and third, they neglect the importance of respecting the original intention behind composite operations for detecting conflicts and constructing a merged model.

With the new requirements in the formal and Agile design and development of softwares shows that the formal methods integrated with Agile methods will produce an adaptive software and software design complexities will be managed easily. Proposed methods for evaluation includes

1. Formal Design
2. Requirements
3. Code generation
4. Code Refactoring
5. Documentation
6. Tool support
7. Distributed development

Formal design

Agile designs are emergent, different approaches and different practices that can exist in agile world include. Formal system design will emerge over time, evolving to fulfill new requirements and take advantage of new ways of integrated formal methods and Agile development. Small changes can be made as part of the solution taking refactoring into account, and for effective design model checking will be done at each instance. After design what needs to be of focus is continuous integration.

A UML class diagram can be used to depict a high-level domain model or a low-level design, not to mention things in between. Use cases can be used to model the essential nature of a process or the detailed system usage description which takes into account architectural decisions.Formal methods in this design will verify the correctness by proving whether the semantics and syntax of the software conforms to the natural language.

Requirements in Agile modeling

High level requirements are always needed for Agile modeling **.**The critical aspects in requirement will require modeling as requirement evolve throughout the model and it needs to give enough estimates of the requirements being sought. Modeling the steps will require successive iterations after viewing the architecture, this will help us to detect the error and a test-driven method, through which those errors can be overcome at all times if they appear later this reduces chances of wrong validation if the requirement is represented correctly with

agility. Agility can understand requirements if formal requirements can be in place when needed.

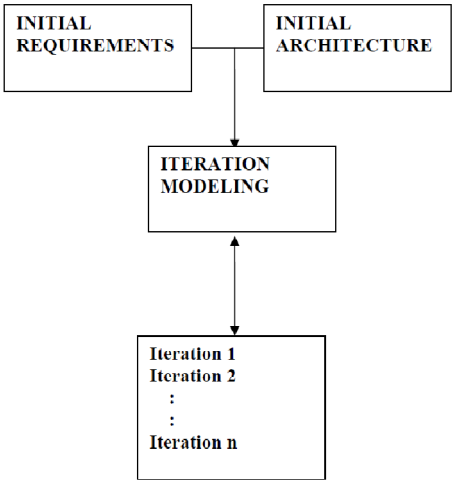

Figure 2. Successive Agile requirements.

Code Generation

In formal methods code generation should transform the whole event based architecture into a sequential program, in code languages; formal definition should be unaltered when the details change. Programming languages should evaluate its arguments first which forms a sequence. A characteristic feature of a program in any programming language shows that if it does not manage to generate an efficient code or any fragment of that program can be executed later. Taking example of a top down approach non-terminating program is represented as true.

Majority of software systems are described in high-level model or programming languages. The runtime behavior, however, is controlled by the compiled code. If the software is in critical systems, it is of great importance that static analyses and formal methods can be applied on the source code level, because this level is more abstract and better suited for such techniques. However, the analysis results can only be carried over to the machine code level, if we can establish the correctness of the compilation. Thus, compilation correctness is essential to close the formalization chain from high-level formal methods to the machine-level.

Code Refactoring

Characterization of formal and Agile methods should restructure the code of the body by formally breaking each aspect into smaller pieces to capture the external behaviors. The developers will change code to improve performance (Paul et al., 2009), The target to berefactored is instinctively refactored to specifically facilitate other processes like verification and validation. After refactoring, the target program is documented with low-level interpretation, and a specification is extracted instinctively. The semantics of the refactored program are equivalent to those of the original program, that the code conforms to the annotations, and that the extracted specification implies the program's original specification constitute the verification argument. This is always applicable to other types of properties such as those that concern the design and maintenance of software systems.

ISSN-L: 2223-9553, ISSN: 2223-9944
Vol. 4 No. 1 January 2013

Tool Support

Tool support for model driven engineering should preferably be integrated into agile development for software design environments with formal methods. Tool support has become a key reference model for software specifications in the world of model-driven development, to bypass the complexities, agile and formal methods can rely on each other and the two can have a suggestion to implement a model driven way of development.

A further important facet of tool support has to do with routine and scalability. It is evident that one can have a possibility to come up with model driven tools that scale up in designing software.

Distributed Development

The immediate task of taking a step in developing huge software systems in a model driven manner gives an emphasis that it would always be very easy, due to the fact that several development teams might be involved. In this case, it would be advantageous if the model could be subdivided into several parts that could be developed in a distributed way. Taking into Consideration software design and this setting, model essentials from unlike sub models might be involved. Thus, several distributed designs and steps have to be performed and potentially harmonized if they involve frequent model parts.

DOCUMENTATION PRACTICES

Majority of software at present comes with wide-ranging documentation. The way components interact helps create some desirable properties and then in the end this should meet their requirements. Using Z notation as a way of documenting the quality of the software can be changed given that formal methods based on elementary mathematics can produce precise documentation because all the information produced at this level is structured well (Spivey, 1992). As formal methods are expected to be a normal practice in software engineering integrating them with agile process can increase ways of developing and designing software system. The Z notation for documentation specifies certain functionalities which represents some rigorous mathematics so that programmers with experience in mathematics could easily have to understand the quality and reliability of documentation (Spivey, 1992).

All the way through the process of understanding documentation and the parameters or indenture that is directing the Formal Requirements Documentation. Spivey (1992) findings can be done on the regulations that rarely dictate how or when to capture the requirements, the level of traceability detail required or directive formal reviews. Given these facts, there is noticeably still bounty of scope for agile methods contained by a formal framework. It is also essential to get an appreciative way of which preferred agile requirements mechanism which will comprise the formal requirements documentation, and determine if there is a synergy between the preferred agile requirement artifacts and the dogmatic constraints.

ARE FORMAL METHODS READY TO HOUSE AGILITY?

Existing complex computer based problems can always be approached by formal methods. Combing Formal methods are considered as Orthogonal. Gruner (2009/2010) views include formal and Agile which behaves like water and Oil but by incorporating Agile into formal methods it is possible to use these mathematical entities to do modeling, examination and scrutiny at different levels and phases. This can improve the chances of combining the two. As formal methods are regarded an expensive way of developing software, if agile development methods can be incorporated inside this formal analysis using extreme

programming software complexities can easily be managed. Agile Methods have the implementation of the principles of the incline production in software development. Therefore, Agile Methods focus on continuous process improvement through the identification and the removal of waste, whatever does not add value for the customer.

AGILE METHODOLOGIES

What Is Agile?

Agile methodology is a loom to project management, classically used in software development. It helps teams respond to the changeability of building software through incremental, iterative work cadences, known as sprints. Agile methodology has been inspired it: waterfall, or traditional sequential development. Agile management develops big effects on the following variables during the development process

Cost

Monetary impacts and the effort to be put into the system.

Schedule

Impacts as timeline is changed.

Requirements

The scope of the work that needs to be done can be increased or decreased to affect the project.

Quality

Unnecessary activities decreases quality.

For the reason that software development is often well thought-out as a sequential, linear process, middle and upper management habitually assumes that all four of these factors can be directed to the development group

FORMAL METHODOLOGIES.

Formal methods relies on the use of meta-mathematical approach to solve software engineering problems at different levels of specification and design, correctness, basically they are mathematical practices, often supported by tools, for emergent software's and hardware systems.

It is the use of thoughts and techniques from mathematics and formal logic to stipulate and reason about computing systems to enhance design assurance and purge defects.

Formal Methods tools allow wide-ranging analysis of requirements and design and a finished examination of system behavior, including liability conditions.

Benefits of MDE formal methods

- Early detection of errors and defects
- Correctness
- Product correctness

Using Agile methods to build formal methods.

Agile methods can always rely upon some parallel software development. A number of methods work to achieve this. With regard to requirements, feedback on how well the methods meet these requirements has to be analyzed making them ready for this task. In

ISSN-L: 2223-9553, ISSN: 2223-9944
Vol. 4 No. 1 January 2013

Agile Formality and Agile development methods are viewed as entities that evolve from real applications.

CONCLUSION AND FUTURE THOUGHTS

In trying to merge formal and Agile methods, human factors should be taken seriously to make formal methods research an acceptance as they are suitable for integration with agile methodologies of software development. This is due the fact that agile formal methods can be thought of as a rigorous and a practical method of development even thought the two are considered to be orthogonal.

REFERENCES

Paul, P. Boca., Jonathan, P. Bowen., Black, S., Gorman, J, & Hinchey, M. (2009). Formal Versus Agile*: Survival of the Fittest? IEEE Computer,* 42(9):37–45.

Black, Sue., Fitzgerald, J. & Wolff, S. (2010). Are Formal Methods Ready for Agility? A Reality Check. 13-25

Jackson, D & Wing, J. (1996). *Lightweight Formal Methods. IEEE Computer*, 29(4):22–23, April

Gruner, S. (2009 / 2010). Workshop on Formal and Agile Methods – Editorial Preface and Foreword. *Innovations in Systems and Software Engineering,* 6, pp. 135-136, Springer-Verlag,

Bianco, V.d., Stosic, & Kiniry, J. (2010). *Agile Formality: A"Mole" of Software Engineering Practices.*

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. (2002). Agile software development methods Review and analysis. *Technical Report 478,* VTT Technical Research Centre of Finland,

Gruner, S. & Rumpe, B. (2010). 2ndWorkshop on Formal Methods and Agile Methods.

Edmund, M., Clarke., Jeannette, M. & Wing. (1996). Formal Methods: *State of the Art and Future Directions September*

Paul, J., Kuzmina, N, Gamboa, R. & Caldwell, J. Toward a Formal Evaluation of Refactorings, Proceedings of The Sixth NASA Langley Formal Methods Workshop, p.33–35

Douglas et al., (2006). *Work Model driven Engineering*. IEEE Computer Society. *Computer* Feb: 25-31

Rumpe, B. (2004). Agile modeling with the uml. In: RISSEF. Volume 2941 of *Lecture Notes in Computer Science.* Springer 297–309.

Fleurey et al., (2009) Qualifying input test data for model transformations. *Software and Systems Modeling*, 8(2), 185–203.

Object Management Group, (2008). Software & Systems Process Engineering Metamodel SPEM 2.0. http://www.omg.org/specs/

Vincent Mahé Benoît Combemale, and Juan Cadavid (2009). Crossing Model Driven Engineering and Agility Preliminary Thought onm Benefits and Challenges. *Software and Systems Modeling* 8(2): 185-203

Eleftherakis, G. & Anthony, J. (2003). Cowling. *An Agile Formal Development Methodology.* Proceedings of the 1st South-East European Workshop on Formal Methods, SEEFM'03 November 2003, Thessaloniki, Greece

Overdick, H., Puhlmann, F. & Weske, M. (2005). Towards a Formal Model for Agile Service Discovery and Integration. Available on http://bpt.hpi.uni-potsdam.de/pub/Public/MathiasWeske/ICSOC2005.pdf

Spivey et al. (1992). *The Z-Notation: A Reference Manual.* Englewood: Prentice Hall