ISSN- 2394-5125 VOL 04, ISSUE 06, 2017

DEVELOPMENT IN SOLVING SOFTWARE COMPLEXITY

[1]Dr.D.Sumathi, [2]Mr.P.V Ramana Murthy,[3] Ms.Sarala Sandhya Rani

[1]Professor,[2]Associate Professor, [3]Assistant Professor,Dept. of CSE,
Malla Reddy Engineering College (Autonomous), Secunderabad, Telangana State

ABSTRACT: Software development process is becoming increasingly challenging and managing software product becomes way too expensive. On the other side the expensive product could not be ported easily across technologies to keep up the phase in changes in technology. This paper highlights the issues in software development and software maintenance and possible solution using Model Driven Development MDD.

INTRODUCTION

Software engineering has evolved greatly since 1960s and taken various paths and corrections in maturing itself into an engineering discipline. In 1960s the problem faced by the developers is to rewrite the software for the new hardware getting released in quick succession. In 1970s and 1980s the major issue is the turnaround time,
i.e. the time taken for the completion of the job. For last ten years solving the needs like reducing complexity, raising the level of abstraction and other needs of the software industry was the main focus of researchers. The cost of software maintenance has grown more than twice in comparison to the development cost. In 1995 most of the project implemented and operation were not considered successful. With respect to the schedule most of the large projects takes at least additional 50% or more slippage in schedule. In a large project 75% of the software is not usable for many reasons. In last 50 years software development was performed with mathematical foundation to solve business and technology problems. There was very little focus on understanding how human solve a problem, how to apply knowledge of the human to the system. This idea is the core concept of artificial intelligence but it was lying in the research and very little industries capitalized it. The Multi-Agent system which is derived from distributed artificial intelligence has methodologies and process to influence the software industry, due to the lack of clarity and lack of technical tools and technology the vast majority of software industry could not able to capitalize the benefits of the technology.

ISSN- 2394-5125 VOL 04, ISSUE 06, 2017

CHALLENGES OF THE EXISTING SOFTWARE DEVELOPMENT

1. Solving complex requirements – the ability to quickly solve complex requirements and to have a more flexible architecture to adapt to any changes faster.
2. Reducing the time to market – it is one of the main challenge of the product development. The expectationis in no time or very less time the product should hit the software market before the competitor product hits the market.
3. Faster change in requirement – generally the application grows faster in requirement once the application is available for the user to work with. These requirement changes would be expected in a faster phase as the user already started using the application.
4. Lowering the cost of production and maintenance – this is one of the main challenge in the Success of the software. Generally the maintenance cost would be more than double in comparison to the production cost. Software development needs methodologies to reduce both the development cost and the maintenance cost as well.
5. Reducing the gap between requirement and application – the need for good tools and Ubiquitous languageto make domain expert participate in the development project in order to reduce the gap between requirement and application.
6. Reliability even with unpredictable changes- This is one of the highly expected demands of the customer; the design should be robust enough to accept possible changes of unpredictable nature.
7. Scalability even under huge payload – now generally the applications are deployed in the cloud or at least distributed through the internet. The need for application to adapt to various time zones is must.
8. Security – This is one of the huge challenges in message communication. With the growing threat in the internet and the growth in the distributed nature of the application it is highly challenging and an uncompromised goal for any application.

MODEL DRIVEN DEVELOPMENT

Model driven development (MDD) is one of the most prominent model based Technology for knowledge management. The popular model driven technologies are model driven architecture (MDA) and software factories of Microsoft. It works on higher abstraction level than traditional programming languages. Models tend to simplify the actual

ISSN- 2394-5125 VOL 04, ISSUE 06, 2017

complexity of the code. Using models complex functionalities could be developed in the same time. It is cost effective by making development not only simpler but also reducing the time to market. It helps build application faster. It increases the understanding of the domain and directly helps reduce the time spent by the developers reading multiple lines of code to make changes for the requested change in requirement. It helps produce high quality software using proven code template in the automation process. It reduces the overall error and hence reduces the testing time. It results in the software which is less sensitive to the people involved reducing the application dependency to the skill of the people. It automatically provides separation of concerns; it provides clarity for domain experts and technical experts. It makes the developers concentrate more on non-repetitive technical tasks and the domain experts in building domain knowledge. It forces design to plug-in the business rules dynamically and enables application to be independent of business requirements. It enforces architecture and design principles forcibly than any other development model. The code generated would be always in compliance with the development guidelines and functional design guidelines. It has a stronger connectivity with Domain Driven Design (DDD).

MDD naturally leads to produce high quality software, if a proven code is used in the automation process. Executing the transformation engine on the model results in the code required, so the quality of the application directly depends on the quality of the code produced during the automation. It greatly reduces the variability in the code which is the resultant if developed by multiple developers. So instead of coding completely, the code with all best practices could be first handwritten and after testing it thoroughly the templates could be prepared on the tested code and generated templates could be used for code generation.

MDD results in the software which is less sensitive to the people involved. The need for the quality personal is confined to the external frameworks exists in the application other the MDD models.MDD avoids the major need of documentation and reduces the burden of keeping the document up-to-date. The model as a key documentation and it is always up-to-date since it is directly mapped to the application artifacts. Since it is built with ubiquitous language it is readable both by technical and domain users. Finally MDD keep the organization focus on business problems rather than technology.

ISSN- 2394-5125 VOL 04, ISSUE 06, 2017

All the above said features consolidate to specify this feature. Like any other technical artifacts the model which is the knowledge artifact could be backed up and could be preserved for many years

Designed to solve domain problems – domain is generally seen as requirements, even though it is true, they are not just simple requirements. Domain has various notations, specification, business rules and various others. When communicating to the domain experts they would be specific on the key areas where careful attention needs to be brought in. But the rest of the entirety of the domain will be in their languages used. So capturing their language is the more fundamental and basic thing to be done. This thesis primarily works on the modeling language to capture the essence of the domain.

Reduction in need for higher skill level – skill level is one of the rare resources in any domain. Generally in any organization the knowledge of the domain lives with its people. So losing a people is the loss for the knowledge available in the organization. To find alternative people to have similar knowledge would be highly competitive currently. Hence capturing the knowledge of skill level is one of the primary aims of this research. DSL is used primarily to capture the knowledge of the organization in models. The captured knowledge could live for many years across several generations of technologies. This helps the people inorganization to update their knowledgeand uses very less effort of people in training other people. In overall it reduces the need for the higher skill level by equipping the existing people.

Maintenance and evolution of models – as discussed earlier models have scope beyond its usage in implementing software for the organization. It primarily becomes a knowledge source. The domain experts could gradually configure the models one at a time and slowly these models evolve and act as a repository of knowledge for the organization. This knowledge could exist for generations to serve across technologies. The need for change in technology with the modeling language would not affect the knowledge captured in the models. Various migration technologies could able to assist porting the existing knowledge source to next generation technologies. Using models reduces the overall complexity and increases the reusability of models and derivatives of the models. This way the overall time spent on maintain software would reduce highly.

Raising the level of abstraction – the usage of models raises the level of abstraction. It

ISSN- 2394-5125 VOL 04, ISSUE 06, 2017

works on higher level of abstraction than traditional programming languages. The higher view point provided by the models actually elevates the abstraction level; the models tend to simplify the actual complexity of the code.

Separation of Concern or Modularity – multi agent programming helps achieveabstract integration of components instead of concrete integration. Their distributed nature, features like adding run time behaviors andtheir reasoning to understand the environment and adapt itself to the environment redefines the way separation is generally thought through. The framework developed for software simulation provides the flexibility for the software to have facilities to configure behavior in the runtime using configuration or in the process ofconfiguring itself using the rules provided in runtime.

CONCLUSION

Model Driven Development comes as a solution for solving software complexity. The models could be ported across technologies and increases reusability and reduces time to market. This way the product could be managed to keep up the changes in the technology. Even though MDD has various advantages, if the model size increases, managing models of larger models will become highly challenging. Further study needs to explore the solution for the model size issue mentioned above.

REFERENCES

[1] J. Greenfield and K. Short, Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools, John Wiley and Sons, 2004.

[2] P. C. Smolik, Mambo Metamodeling Environment, A dissertation submitted in partial fulfillment of the requirements for the degree of doctor of philosophy, Brno University of Technology, Czech Republic, 2006.

[3] OMG Trademarks, http://www.omg.org/legal/tm_list.htm, January 2007.

[4] S. Cook, Domain Specific Modeling and Model Driven Architecture, MDA Journal, January 2004,