International Journal of Computer Applications (0975 – 8887)
Volume 65– No.19, March 2013

Software Developing with Agile Methods and Combination of Architecture

Mehdi Yaghoubi
Department of Computer Sciences,
Faculty of Sciences,
Golestan University, Gorgan, Golestan, Iran

Manoochehr Babanezhad
Department of Statistics,
Faculty of Sciences,
Golestan University, Gorgan, Golestan, Iran

ABSTRACT

Over the past decades the service oriented architecture is a viewpoint of software architecture in which using Loosely Coupled services to support business processes is suggested. Nowadays most of the software has been established based on the service designing and implementing. Due to this, this article examines architecture and its role in the agile methods of software development. On the other hand, the agile methods of software development, in comparison to the other methods have gained more customers. This might be because it tries to control the changes rapidly and obtains business profit. In the past, it was believed that the architecture and agility were two completely different concepts and cannot be combined, because in the architecture the quality is important and all the requirements should be predicted previously. In agility, accepting new requirements and adapting them are important but today, agility in the domain of architecture is growing rapidly. Some experts believe that they should include architecture in the process of agile improvement. However the others focus on making the architecture working cycle more agile. We also show that the method of light software development (LSD) is one of the methods that meets to the combination of service oriented architecture and agility.

Keywords

Agile methods, Software development, Software Architecture, Service Oriented Architecture.

1. INTRODUCTION

Service oriented architecture (SOA) is one of the architecture methods that provide business operations Integration by using services [1, 2]. The services have the least dependencies with each other. So changing the service is done easily and the system magnifying would be also done in the best way. We can readily add some elements to software or eliminate some elements of it[2, 3,4]. As a result when the changes are high in the environment and there appear new requirements in the business and the enterprise needs to perform some changes, using this architecture would be helpful. On the other hand the agile methods of software development have been highly used in many projects because of their most advantages that have in comparison to the heavy weighted methods. Adopting the changes is one the principles of the agile methods that we would be introduced by these principles below. So using SOA in the agile methods could be helpful and profitable. Of course this is just one of the advantages of SOA [5, 6]. Indeed SOA is one of the architectures that are applied by the aim of decreasing information technology costs and improvement of agility in the enterprises [7]. In the past it was believed that architecture and agility were two concepts completely different and on the opposite side of each other and could not be combined. In some agile methods even the role of the architect has not been considered. For example the roles existing in the methodology of XP that is one of the famous agile methods include: programmer, customer, Tester, Tracker, Coach, big boss and consultant. The agile methods of software development do generally lack a stage named architecture. In heavy weighted methods, the place of architecture is after Requirements analysis and before designing. But in agile methods Requirements analysis and designing are considered as subsets of modeling order. Therefore one of the possible alternatives to combine architecture and agility is to contain architecture in the process of agile development. Agile model driven development (AMDD) is one of the main methods of agility development and meanwhile it has considered a place for model driven architecture in its working cycle. The other trend is agility making of architecture working cycle and especially architecture documentation. Regarding these trends, the researchers have offered methods to combine the architecture and agility.

In this article at first in second section of the article we will discuss agile methods. Third section contributes to the architecture and agility. This section is contained two parts named "architecture in the methods of agility development" and "agility in the working cycle of the architecture". In the following the relationship of service oriented architecture and agility has been examined and an approach named light software development (LSD) would be explained that is both based on service oriented architecture and use the agility development principles. The last part of the article is allocated to the conclusions.

2. AGILE METHODS OF SOFTWARE DEVELOPMENT

The agility subject in software industry dates back to 2001. In that year, 17 out of researchers such as Alistair Cockburn, Martin Fowler, Kent beck, Bob Martin meet each other to rest and go skiing in addition to discuss. The result of this meeting was a manifesto that became famous to agile software development manifesto. This manifesto is as follows [8]:

• Individuals and interactions over processes and tools

• Working software over comprehensive documentation

• Customer collaboration over contract negotiation

• Responding to change over following a plan

Since the publication of agility manifesto on, several methodologies in the field of agile methods of software development have been offered by different enterprises and individuals. AUP, AMDD, FDD, XP, Scrum are samples of these methodologies. All these methodologies attempt to simplify the process of software development. For example the process of agile development of AUP is simplified version of RUP process that was offered by Scott Ambler in 2005. This process develops a simple and comprehensible approach to develop application software using agile concepts and techniques and meanwhile it follows up to the principles of RUP development. Figure 1 shows the life cycle of the agile development of AUP that similar to RUP life cycle of development process has two different dimensions. First dimension (vertical axis) shows work flow or disciplines. The second dimension (horizontal axis) shows development cycle of the software during the time that is composed of four stages of inception, elaboration, construction and translation [9]. The first difference that appears herein in comparison to RUP development process is that its disciplines have been changed and the model order in AUP has been replaced by three disciplines of business modeling, analysis and design discipline and requirements disciplines in RUP. The second difference is that configuration and change management discipline in RUP is here just as configuration management discipline. Because in the process of agile development, the activities related to change management is considered as part of Requirements management that is placed within the model discipline [9].

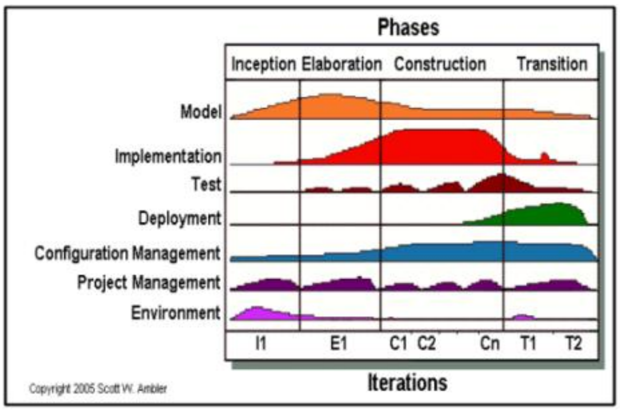


Fig 1: the life cycle of AUP agile development process

3. ARCHITECTURE AND AGILITY

Agility is the ability of establishing and answering the changes in order to obtain profile in agitated business environment and close relationship with the change issue. A change that arises from the business environment changes and entails the enterprises to oppose it. The concept of agility spreads in many areas like production and development of software, enterprises and also architecture rapidly. The agile architecture and agile enterprise architecture are of those concepts that are offered in response to the need to dynamic reusability in architecture and provide the ability of controlling and managing the changes for architects [10]. IBM cooperation researchers, in their studies on this field [11,12] focus on the agility combination with architecture. In addition, they assessment the agile architecture benefits and the enterprise readiness to adopt with this kind of architecture. Also time, Effort and costs have been considered. In some articles that are published by IEEE [1,4,6] architecture and agility and work fields of each of them have been studied and all the authors have asked the combine of architecture with agility in some way. Some of them believe that architecture or the similar phase with it should be contained in the agile development process. In the case that the others suggest this idea that architecture work cycle should be implemented using agile methods.

3.1 Architecture In Agile Development Methods

As was said, the agile methods of software development do generally lack the stages named architecture; because the position of the architecture is after requirement analysis and before designing. But, in the agile methods, requirement analysis and design are considered as subset of modeling discipline. So one of the possible alternatives to combine architecture and agility, use architecture stage or the similar one in the software agile development methods. For example, AMDD is one of the main methods of the agile development that works based on model driven techniques and meanwhile considers a place for architecture in its work cycle [13]. Figure 2 shows the life cycle of the agile model driven development. In the first repetition (iteration 0), the program should be organized and the correct route should be selected .In this stage the Requirements analysis and initial architecture is done .Then in the other Iterations, modeling, the model storming and test driven development would be done . To more confidence, a review could be done after each iteration [14, 15].

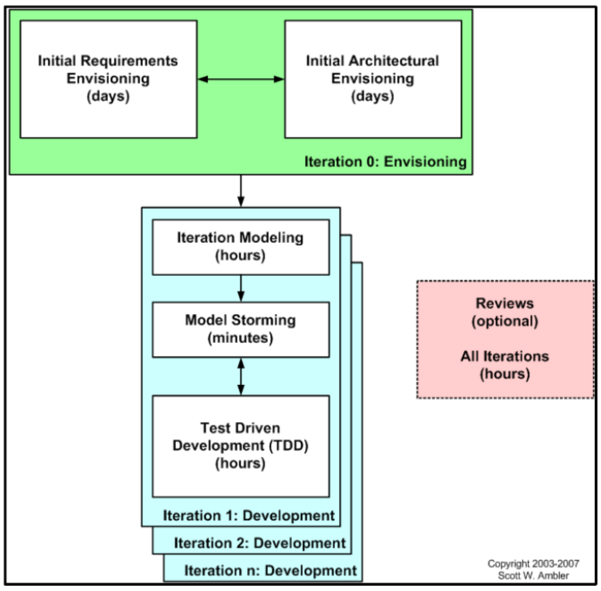
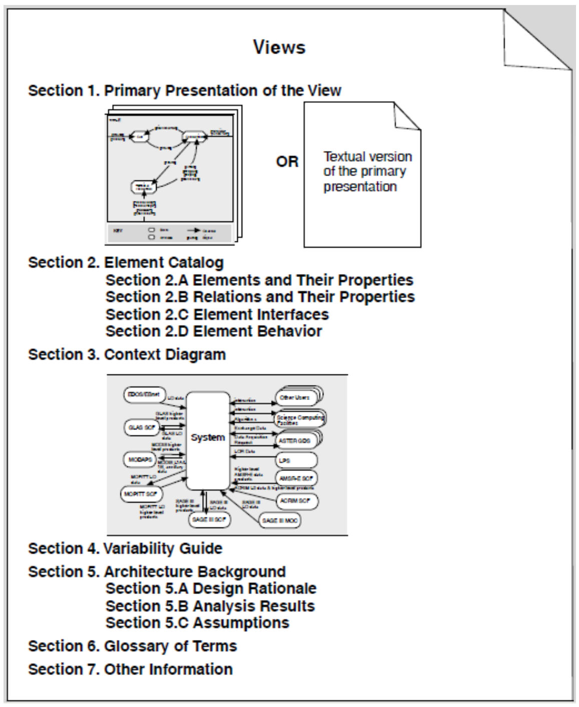

Initial Requirements Envisioning (days)
Initial Architectural Envisioning (days)
Iteration 0: Envisioning
Iteration Modeling (hours)
Model Storming (minutes)
Test Driven Development (TDD) (hours)
Iteration 1: Development
Iteration 2: Development
Iteration n: Development
Reviews (optional)
All Iterations (hours)
Copyright 2003-2007 Scott W. Ambler

Fig 2:The life cycle of the agile model driven development

3.2 The Agility in Architecture Work Cycle

Researchers at the SEI and the Carnegie Mellon School of Computer Science set out to answer the question: "How should you document architecture so that others can successfully use it, maintain it, and build a system from it?" The result of that work is an approach that is called "Views and Beyond" or "V&B".

The basic principle of V&B is that documenting software architecture involves documenting the relevant views, and then documenting the information that applies to more than one view. V&B includes a method for choosing the relevant views based on the structures that are inherent in the software architecture and on the needs and concerns of the architecture documentation's stakeholders. Figure 3 Shows the V&B Outline for Documenting a View.



Views

Section 1. Primary Presentation of the View

OR

Textual version of the primary presentation

Section 2. Element Catalog
Section 2.A Elements and Their Properties
Section 2.B Relations and Their Properties
Section 2.C Element Interfaces
Section 2.D Element Behavior

Section 3. Context Diagram

Section 4. Variability Guide

Section 5. Architecture Background
Section 5.A Design Rationale
Section 5.B Analysis Results
Section 5.C Assumptions

Section 6. Glossary of Terms

Section 7. Other Information

Fig 3: V&B Outline for Documenting a View

Architecture architectural documentation is a step of work cycle that is very time-consuming in Heavyweight methodology and are created large amount of documentation. So, one the possible trends in combination the agility with the architecture can be agility of architectural documentation step. In a report that was published by a group of researchers of software engineering institute in Carnegie-Melon University, use views and beyond ("V&B) approach to document software architecture in the agile methods [3]

SEI suggested solution to generate architecture documents using the agile principles are offered composed of stages that follow [3]:

1. Begin by creating a skeleton document for comprehensive view-based software architecture

Document using the standard organization schemes shown in Figures 5 and 6. However,

Start with the outline only and leave the sections filled in initially with "to be

determined."

2. Using the view selection scheme of the V&B approach, decide which architectural views you would want to produce, given enough resources. Choosing a view at this point does not obligate you to document it, but rather serves as a confirmation that there is a stakeholder community who will find information in that view useful, no matter how it is communicated. Choosing a view identifies a family of design decisions that the architect needs to resolve and be able to express. Add outlines for the chosen views to the outline you created in Step 1.

3. Annotate each section of the outline with a list of the stakeholders who should find the

information it contains of benefit. Don't forget stakeholders who might not have joined

the project yet, especially new hires, the maintenance staff, and successors to the current

architect(s).

4. For sections that have an important stakeholder constituency and that you can fill in

quickly using material at hand, do so. For example, a system overview available from

other sources can be put to use easily. Or, the whiteboard sketches that agile methods prefer

can be captured and put into the appropriate place(s) in the documentation skeleton.

5. Prioritize the completion of the remaining sections:

• If a section's constituency includes stakeholders for whom face-to-face conversation is

impractical or impossible (e.g., maintainers in an as-yet-unidentified organization), that

section will need to be filled in. If it includes only such stakeholders, its completion can be

deferred until the conclusion of the project's development phase.

• If a section's constituency includes only stakeholders for whom face-to-face conversation

is practical and preferred, it may not need to be filled in. However, the architect may prefer

filling it in to repeatedly answering the same questions about it. If a question about information in a particular section is asked, you can capture the question and answer it in that section. Thus, optional sections can become a list of frequently asked questions (FAQs)

about the architecture that can be captured at a minimal cost.

• If a section's constituency includes both close-in and far-off constituents, try a combination

of the approaches. Capture an FAQ list and convert it to a form more appropriate for

archival purposes as time and resources permit.

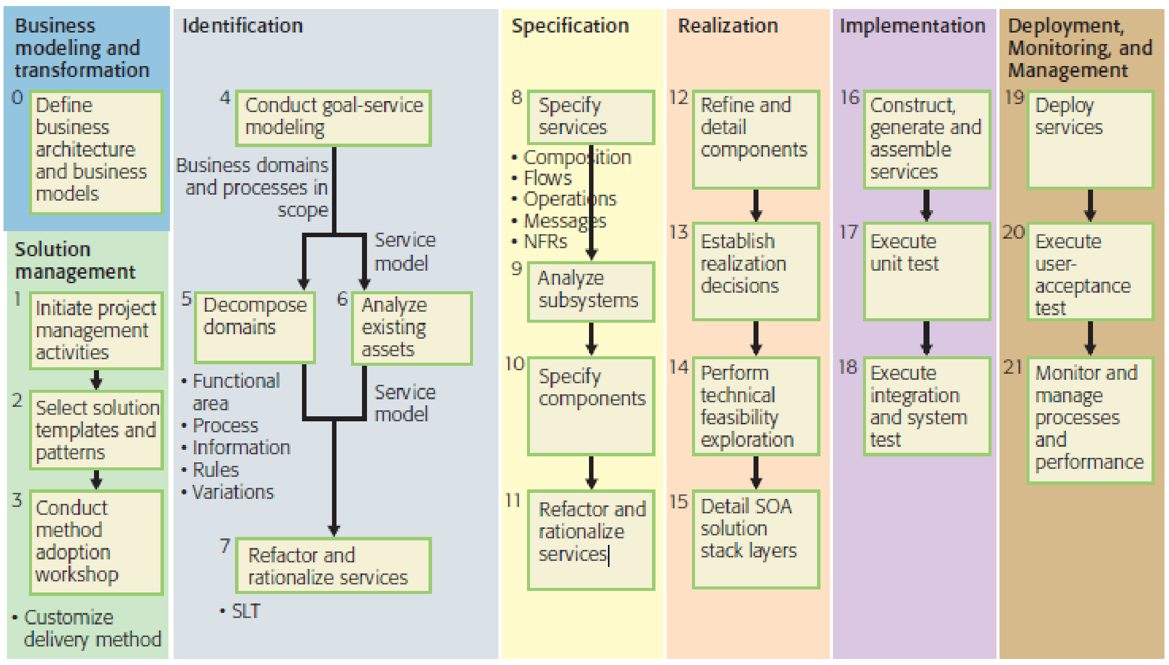

Business modeling and transformation
0 Define business architecture and business models
Solution management
1 Initiate project management activities
2 Select solution templates and patterns
3 Conduct method adoption workshop
• Customize delivery method
Identification
4 Conduct goal-service modeling
Business domains and processes in scope
Service model
5 Decompose domains
6 Analyze existing assets
• Functional area
• Process
• Information
• Rules
• Variations
Service model
7 Refactor and rationalize services
• SLT
Specification
8 Specify services
• Composition
• Flows
• Operations
• Messages
• NFRs
9 Analyze subsystems
10 Specify components
11 Refactor and rationalize services
Realization
12 Refine and detail components
13 Establish realization decisions
14 Perform technical feasibility exploration
15 Detail SOA solution stack layers
Implementation
16 Construct, generate and assemble services
17 Execute unit test
18 Execute integration and system test
Deployment, Monitoring, and Management
19 Deploy services
20 Execute user-acceptance test
21 Monitor and manage processes and performance

Fig 4: SOMA life-cycle high-level flow

4. COMBINATION OF SERVICE ORIENTED ARCHITECTURE AND AGILE METHOD

The service oriented architecture provides an especial method to overview a system which the ability of adapting ,i.e., the ability of responding to the changes and new requirements are being focused. So it is clear that the agile methods of software are the best choice to develop these kinds of systems [5]. So, in order to provide a adoptable operational and flexible environment in service-oriented architecture, any method used should have high degree of agility .One of these methods that was initiated by Mary and Tom Popendik is LSD that is based on service orientation and agility. LSD has 7 principles as follows [16]:

Principle 1: Optimize the whole

Principle 2: Eliminate waste.

Principle 3: Build quality in.

Principle 4: Learn constantly.

Principle 5: Deliver fast.

Principle 6: Engage everyone.

Principle 7: keep getting better.

By examining the above principles, an operable solution of service-oriented architecture has been introduced by IBM experts with the supervision of Ali.Arsajani, Ph.D by offering an approach called SOMA [17]. This methodology is composed of 7 disciplines of "business modeling and transformation", "solution management", "identification", "specification", "Realization", "implementation", " deployment, monitoring and management" (figure 4) [2].

In addition, in the other researches performed by IBM, a new concept called service-oriented agility has been mentioned in which a combination of software development principles is offered by developing service-oriented architecture [18, 19]. The result obtained from the researches indicates that service-oriented architecture could enjoy documented principles of agile development and specially LSD method. The combination of service oriented architecture and agility result from the common base that both attempt to adapt with changes. In other words, the agile development methods with service oriented architecture has natural difference and they cannot be mixed with each other like water and oil ; but LSD can be used as a combining interface of these two technologies[19].

5. CONCLUSIONS

In this article, we first talked about service oriented architecture and its advantages in to comparison the other traditional method. In the following, the issue of software development agile methods, properties and priorities of the agile methods in comparison to the other software development methods is proposed. We said that the agile method generally lacks architectural stage. In contrary to the previous experts who thought architecture and agility could not be combined, two main trend are presented in the field of combining architecture and agility, one of which is to use architecture in the agile methods and the other is to make agile the architecture work cycle. For example, in the first trend, AMDD could be named that is one of the main agile development methods and meanwhile consider a position for model driven architecture in its work cycle. Also we said that the second trend searches to make agile the architecture specially the architecture documentation. Because the architecture documentation is one of the architecture work cycle stages that is very time consuming in heavy weighted methodologies and large amount of documents. In the last section of the article, the relationship between service oriented architecture and agility is examined. To provide an adaptable and flexible operation environment in service-oriented architecture, any method used should have high degree of agility. One of these methods is LSD that is both based on service-oriented architecture and use the agile development principles. The combination of service oriented architecture and agility is based on this common principle that both try to adapt with the change. In fact, the service-oriented architecture conforms the agile development principles well. In other words, the agile development methods are naturally different from SOA and they cannot be combined; but LSD method could be used as an interface combining these two technologies. Any way regarding the role the architecture plays in the success of a software system and regarding the fact that today most of successful software use SOA, the advocates of the agile methods pay special attention to the combination of service-oriented architecture and agile methods of software development. The future we will see other methods representation in this view.

6. ACKNOWLEDGMENTS

This article was supported by the grant of Golestan University. We would like to thank Golestan University to support this project.

7. REFERENCES

[1] Abrahamsson, P., & Babar, M., & Krutchen, P., (2010) *"Agility and Architecture : Can They Coexist?"*, IEEE Software, Vol. 27, No. 2, pp. 16-22.

[2] Arsanjani, A., & Ghosh, S., & Allam, A., & Abdollah, T. & Ganapathy, S. & Holley, K. (2008) "SOMA:A method for developing service-oriented solutions", IBM SYSTEMS JOURNAL, VOL 47, NO 3, pp.377 – 396.

[3] Clements, P., & Ivers, J., & Little, R., & Nord, R., & Stafford, J., (2003) ,*"Documenting Software Architectures in an Agile World"*, Carnegie Mellon University, Software Engineering Institute, CMU/SEI-2003-TN-023, Pittsburgh, PA.

[4] Erdogmus, H. (2009), *"Architecture Meets Agility"*, IEEE Software, Vol. 26, No. 5, pp. 2-4.

[5] Krogdahl, P., & Luef, G., & Steindl, C. (2005), *"Service-Oriented Agility: An Initial Analysis for the Use of Agile Methods for SOA Development"*, IEEE International Conference on Services Computing (SCC'05), Orlando, FL, USA.

[6] Nord, R.L., & Tomayko, J.E., (2006), *"Software Architecture-Centric Methods and Agile Development"*, IEEE Software, Vol. 23, No. 2, pp. 47-53.

[7] http://www.soaglossary.com

[8] http://www.agilemanifesto.org

[9] http://www.ambysoft.com/unifiedprocess/agileUP.html

[10] Http:///isa.sbu.ac.ir/agile/ index.htm

[11] http://www.ibm.com/developerworks/architecture/library/ararchman1/index.html

[12] http://www.ibm.com/developerworks/architecture/library/ararchman2/index.html

[13] http://www.agilemodeling.com/essays/amdd.htm

[14] http://www.ibm.com/software/solutions/soa/glossary

[15] http://www.enterprisearchitecture.ir

[16] http://www.poppendieck.com

[17] http://www.ibm.com/developerworks/library/ws-soa-design1/

[18] http://www.ibm.com/developerworks/webservices/library/ws-agile1/index.html

[19] [19]http://www.ibm.com/developerworks/webservices/library/ws-agile2.html