Journal of Theoretical and Applied Information Technology
10th August 2015. Vol.78. No.1

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

APPLYING CASE BASED REASONING IN AGILE SOFTWARE DEVELOPMENT

AIMAN TURANI
Associate Prof., Faculty of computer science and Engineering, TAIBAH University, Medina, KSA
E-mail: aimanturani@hotmail.com

ABSTRACT

There is a common misconception among developers who follow Agile development methods, that following formal processes and modeling are unnecessarily and perceived as a waste of effort [1]. The initial intension of Agile was not an anti-methodology movement but rather a balance between processes and production. Nevertheless, the main challenges facing software that are developed using Agile development methods are the risk of higher architectural design mistakes, and the slower transfer rate of knowledge especially when valuable developers and expertise tend to leave their organizations.

Agile advocates itself as a framework based on engaging knowledge workers in affective way. It focuses primary on transferring the tacit type of knowledge within production teams. Yet transferring the explicit type of knowledge is important as well. Reusing past projects' artifacts will have a positive impact on the reduction of costly architectural mistakes and the increasing of the overall learnability, productivity and efficiency of the organization as a whole. Both tacit and explicit knowledge are needed. A hybrid approach that combines both kinds of knowledge is vital and essential especially for teams with diversity skills and knowledge working closely throughout project's development life cycle.

This paper focuses on two main Knowledge Based Management processes that should be applied within Agile development methodologies. The first process focuses on the transformation of tacit knowledge into explicit knowledge using the traditional design models techniques. The second process focuses on usage of Case Based Reasoning systems for facilitating the retrieval and reuse of past projects' solution artifacts. We have selected CBR over other Knowledge Base systems due to its effectiveness in representing solutions for software development domain, which is considered a complex domain, without the need of having a large set of training cases. In this paper we have proposed an ICBR (*Product Backlog* **I**tem **C**ase **B**ased **R**easoning) that could be easily integrated within many Agile development methodologies to effectively disseminate valuable knowledge among organizations' teams and personals. In this paper we have applied ICBR on the Scrum methodology as a proof o concept for facilitating the dissemination of various software artifacts among team's developers to maximize the use of cooperative wisdom and experience found within organizational entity.
Keywords- *Agile development, Knowledge based Management, Scrum, and Case Based Reasoning.*

1. INTRODUCTION

Agile has gained an increasing popularity in software development during last decade [2]. It has increasingly used at various types of projects such as e-commerce, e-services, e-government etc. due to its quickness in responding to unpredictability businesses changes.

Agile is not a single methodology, framework or process. It is a collection of core value statements and principles [3]. Mainly Agile emphasizes the following values: *individuals and interactions* over processes and tools, *working software* over comprehensive documentation, *customer collaboration* over contract negotiation, *responding to change* over following a plan.

Critics of Agile consider it as an anti-methodology movement where process and documentation are not valued. The agile manifesto values working software over comprehensive documentation, but what if developing team's tacit knowledge is insufficient. In addition, without a proper modeling more projects' delays and reworks could happen caused by critical architectural mistakes. Architecture discontinuities for instance, could lead to serious performance and security problems. The amount of rework needed to overcome these types of mistake is substantial where no simple refactoring could fix [4].

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

Architectural mistakes usually happen when teams' main concern is on early working product. Scaling that product in later stages would usually uncover these mistakes.

Adopting more inclusive knowledge management processes and concepts can be used to overcome some of Agile's challenges. For instance, producing explicit knowledge via design models and diagrams would facilitate both team communication and solution reusability. A team member can communicate his thoughts to others effectively by drawing simple diagrams on a data show using, for instance, UML notation. His colleagues can understand his solution faster and elaborate more effectively. Furthermore, other developers could reuse that solution in successive projects. As known, the similarity found between light weight software products, such e-commerce and e-service products, tends to be high. Many functionalities are common across applications within the same domain and reusing design ideas, and software components would lead to a higher efficiency and productivity.

Case-based reasoning offers a considerable potential mean of indexing and retrieving previous project knowledge using simple natural language descriptions.

2. SCRUM METHODOLOGY

In this study we have chosen Scrum methodology as a representative of Agile development methodologies. Scrum is a good example of Agility due to its simplicity and flexibility [5]. The Scrum emphasizes team communication and collaboration, and rapid response to business changes. Scrum software development is mainly evolved via a series of sprints. Each sprint represents a development phase where its duration is from one to four weeks. It usually begins with a concise planning meeting and concludes with a review meeting. Product items are designed, coded, and tested during sprints.

The main roles involved in Scrum are *Product Owner*, *Development Teams*, and *Scrum Master*. *Product Owner* determines what needs to be built in each sprint. He represents customer's requirements and priorities. *Development Teams* build what is needed in each sprint. *Scrum Masters* ensure the sprint process happens smoothly. He acts as a facilitator for both the *Product Owner* and the *Development team*.

A list of all desired work on the project is called Product Backlog. They represent all users' requirements. Product Backlog consists of a list of *items*. They are expressed in such way where each *item* has value to the end users of the product.

3. CASE-BASED REASONING IN KNOWLEDGE MANAGEMENT

Case-Based Reasoning system is a part of knowledge based systems. It is an artificial intelligence based system designed to imitate human problem solving. Usually, when a human is faced with a new problem, he searches his memory and looks for similar past problem and applies its solution to the current problem. So principally, it is reusing old experiences to understand and solve new problems. We have selected CBR over other Knowledge based systems, such as rule based systems or machine learning systems, due to the following key points. Rule-based systems are based on set of rules that need to be obtained by knowledge engineer working with domain experts. These rules are difficult to be formulated within software development domain due to its complexity. Knowledge based systems that are based on artificial intelligent need a large set of training cases, which are difficult to get, to form generalization and identifying the commonalities between a retrieved case and the target case. On the other hand, CBR is effective in getting solutions within rich complex domains with relatively small size set of training cases.

Adapting Case-based reasoning CBR usually consists of the following process [6]:

1. Development of CBR system to store and retrieve cases.
2. Development of Case Library that consists of large amount of cases to enhance the set of retrieved possible solutions.
3. System operation and deployment process
4. Database mining process
5. Management and organizational support process
6. Knowledge transfer process

Journal of Theoretical and Applied Information Technology
10th August 2015. Vol.78. No.1

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

4. ADOPTING CASE-BASED REASONING WITHIN SCRUM FRAMEWORK

In this research there are two main stages applied as shown in Figure 1. The first stage focuses on the transformation of tacit knowledge within the organizational teams into explicit knowledge using Knowledge based management mechanism called Externalization. In this stage teams attempt to rationalize their tacit knowledge and express them into formal models such as UML [7]. The second stage focuses on employing Case Based Reasoning systems for retrieving and reusing past project knowledge.

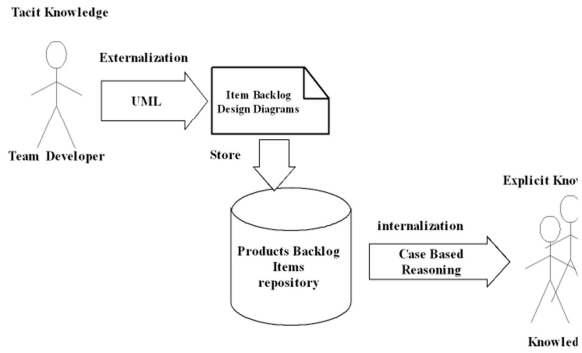

Figure 1: Adopting Case-based Reasoning process

5. EXTERNALIZATION KNOWLEDGE WITH UML

As known, UML is based on well-defined diagrammatic notations. In this paper we focus on two primary benefits of using UML for developing a software application based on Scrum methodology. These benefits are communication and reusability. The use of UML to communicate design ideas within a *Developing Team* during Scrum's sprint planning meeting tends to enhance the flow of knowledge between development team members. This can increase the overall employee's learnability, creativity, and reduces the architectural mistakes. Modeling using UML allows for the production of knowledge based documents that can be reused over several other projects.

UML notations can clearly demonstrate their conceptual contents in a well structured and modular form more than code. In addition, representing and communicating creative design ideas can be instantly done using UML. UML can also be useful in reducing architecture mistakes. It allows a *Developing Team* to deal smoothly with architectural complexity at various levels of abstraction and provoke *team*'s input regarding architecture and design decisions to resolve any potential architectural mistake.

Reusability on the other hand, advocates reusing software solution artifacts (such as UML design diagrams, Use Cases, GUI designs, Database schema, code, etc.) for more productivity and efficiency. In rapid application development methodologies reusing software artifacts at early stage of the software production could lead to faster design decisions with fewer errors. Due to the similarity between same domain projects, for instance e-flight reservation projects, it is normal to find developing teams making the same diagrams over and over again. If these diagrams are stored in a repository, it is most likely other team will reuse them. Reusing software designs is more economic and intuitive than building whole software from a scratch.

6. ITEMS CASE BASED REASONING

During the Sprint Planning meeting, a *Developing Team* collaborates in selecting Sprint backlog item that they intend to work at. Traditionally, items are analyzed to estimate tasks and efforts needed to develop them. In order to promote knowledge externalization, a new task (ICBR) should be carried out within the development of each Product backlog for better effort estimation and knowledge reusability. In this paper, a proposed process of ICBR task consists of the following steps:

1. Search the case library for similar Product backlog item using a backlog item text description.
2. Select and retrieve the most similar past backlog item/s. This involves the comparison of the current item with the retrieved items and somehow ranking them in an order.
3. Estimate the effort needed to develop the current backlog item based on effort made on the retrieved Item.
4. Adapt the design solution of the retrieved item.
5. Add the newly solved backlog item along with its actual effort duration to the case library for future reuse.

Journal of Theoretical and Applied Information Technology
10th August 2015. Vol.78. No.1

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

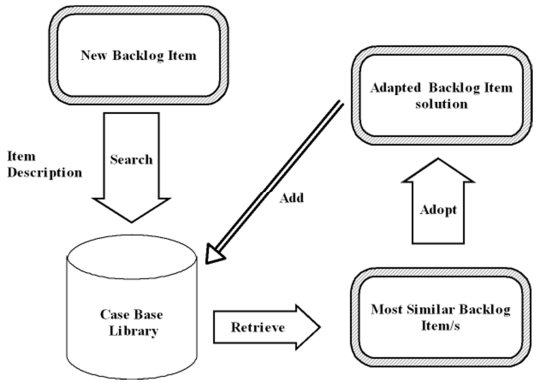

Figure 2: Items Case Based Reasoning

7. SQL SERVER AND SEMANTIC SEARCH

In this research we are using the Semantic Search function within the SQL server [8] as our Case Based Reasoning system due to its capability to store large scale of document and its easiness to use. Most modern search engines, such as Google, are samples of accessing unstructured data using semantic search. Semantic search in SQL Server enables new capabilities that are beyond typical keyword searches. Full-text search enables querying words, while semantic search enables querying the meaning of the document. Semantic Search can enable a user to perform a detailed search into unstructured documents stored in the databases to find any potential similarity with inquired text. It extracts and indexes statistically relevant texts within a document then it uses these texts to identify and index *documents that are related*. Each document has a vector with a vector component of each keyword. The document similarity index uses a cosine similarity algorithm to determine the angle between two vectors. The smaller angle between their keyword vectors indicates more similarity between documents. Documents types that are supported by search using SQL Server can range of many types, including Word, Power Point, PDF, Excel, HTML, and XML.

In this research the main scenario of using SQL server for Semantic Search function have been done according to above listed four steps. Firstly, all past backlog items artifacts are saved in two file tables, the Description file table, and the Solution file table. The Description file table is used to store backlog items description documents. The Solution file table is used to store backlog items solution artifacts and effort duration. Both tables are linked together using a Backlog Item ID. SQL server enables *Team Developers* to search through the Description file table using full-text search and semantic search based on the investigated Backlog Item description. The SQL server then returns all similar items. Secondly, a *Team Developer* browses the retrieved solution of each backlog item. Thirdly, the *Team Developer* can reuse and adapt the design solution of the most similar Backlog Item. Finally, the *Team Developer* adds the newly/updated solved Backlog Item to both file tables.

8. EVALUATION AND FUTURE WORK

A simple evaluation based on a basic questionnaire has been conducted on 27 developing teams' members working on two separate development companies shows encouraging results on using ICBR process in their work over the last 5 months. As table 1 shows that 58% thinks that ICBR has helped them in learning new knowledge from past project.

Table 1: ICBR Usage Questioner Results

	Stro. Agree	Agree	Un Decided	Disagr.
Using ICBR have helped me in learning new knowledge from past project	5	10	7	5
Using ICBR had reduce architectural mistakes	3	9	14	1
Using ICBR had reduced development time	4	9	11	3

These preliminary results are promising and we think the results will show further satisfactory in the future when teams get more used to this new approach. Other quantities assessments are currently used based on a measurement technique, Net hours saved metric, and we are looking toward using other techniques for more comprehensive evaluation. Finally, this works might gain additional importance within virtual developing teams where virtual distant will further weaken the flow of tacit knowledge within these disperse teams.

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

9. CONCLUSION

The main objective behind this paper was to investigate the use of two essential processes within knowledge base management to overcome the limitation and challenges found within the Agile Development Framework. The Externalization process has been used to transfer tacit knowledge into explicit knowledge by using UML. The other knowledge based management process was using Case Based Reasoning for retrieving and reusing past project solutions. We have proposed ICBR process within Scrum methodology for reusing software artifacts and more accurate effort estimation.

Semantic search feature within SQL server has been used as the implementation medium due to its scalability and efficiency. In this paper we have also proposed a general process of using Semantic Search function within SQL to facilitate the transfer of past projects' knowledge based on the description text of the investigated Product Backlog Item.

Adapting ICBR process within Agile methodology would have a positive impact on the Knowledge transfer which maximize the use of wisdom and minimize the mistakes within organizational entity.

ACKNOWLEDGEMENT

We would like to owe thanks to *TAIBAH* University, College of Computer Science and Engineering (CCSE), KSA for supporting this research

REFERENCES

[1] A. Qumer, B. Henderson-Sellers, "A framework to support the evaluation, adoption and improvement of agile methods in practice", *Journal of Systems and Software,* Vol 81, No 11, 2008, pp. 1899–1919.

[2] Torgeir Dingsøyra, Sridhar Nerurc, VenuGopal Balijepallyd, and Nils Brede Moe, "A decade of agile methodologies: Towards explaining agile software development", *Journal of Systems and Software*, Vo 85, No 6, 2012, pp. 1213–1221.

[3] Boehm, B., & Turner, R., Balancing agility and discipline: A guide for the perplexed. 2003, Addison-Wesley Professional.

[4] Lan Cao, Kannan Mohan, Peng X, and Balasubramaniam Ramesh,"A framework for adapting agile development methodologies", *European Journal of Information Systems,* Vol 18, 2009, pp. 332–343.

[5] Cardozo, E., Neto, J. B. F. A., Barza, A., França, A., & da Silva, F. SCRUM and productivity in software projects: a systematic literature review. *In 14th International Conference on Evaluation and Assessment in Software Engineering (EASE),* 2010.

[6] Montani, S., & Jain, L. C., Case-Based Reasoning Systems. *In Successful Case-based Reasoning Applications-2*, Springer Berlin Heidelberg., pp. 1-6, 2012.

[7] Fritz Solms, Dawid Loubser " URDAD as a semi-formal approach to analysis and design," *Innovations in Systems and Software Engineering*, Vol 6, No 1-2, 2010, pp. 155-162.

[8] Microsoft MSDN, Semantic Search (SQL Server), *https://msdn.microsoft.com/en-us/library/gg492075.aspx* (retrieved at 2015).