2011 Ninth International Conference on Software Engineering Research, Management and Applications

Quality Attribute driven Agile Development

Sanghoon Jeon[1], Myungjin Han[1], Eunseok Lee[2], Keun Lee[1]

[1]Memory Divison, Semiconductor Business, Samsung Electronics Company, Hwasung, Korea
{sh47.jeon, mj7.han, gskeun.lee}@samsung.com
[2]Software Engineering Laboratory, SungKyunKwan University, Suwon, Korea
leees@skku.edu

***Abstract*—Agile development methods are being recognized as popular and efficient approaches to the development of software systems that have features such as a short delivery period and unclear requirements. They emphasize customer satisfaction, fast response to changes, and release in less time. According to a recent survey, SCRUM is one of the most popular methods that are currently being used. Some backlogs, especially high-priority backlogs that are functional requirements of customers, are developed repeatedly at each sprint period. Despite the known advantages of SCRUM, however, its backlogs focus only on functional features. Thus, it is difficult to effectively reflect the softwares quality attributes. As known, the failure of a software project is caused by the non-satisfaction not of functional features but of quality attributes, such as performance, usability, and reliability. This paper introduces the ACRUM[1] that is a quality attribute driven agile development method. The main characteristic of the proposed solution is that it is derived from values and practices of SCRUM to be compatible with the SCRUM process and to keep its agility intact. The effect of ACRUM was evaluated through an agile process evaluation checklist and applying it into a commercial project of Samsung Electronics. The results showed that ACRUM is more efficient than the legacy agile development process.**

I. Introduction

Until the end of the 1990s, software projects had been developed by many people and with much expense within a long development period, which is the conventional method of software engineering for such projects. Nowadays, however, software projects are being developed with less expense and within a short development period. Moreover, they are becoming complex and open. Software projects and requirements are dramatically changing with the emergence of dynamic societies or market situations. The Agile method, which is based on the clear premise that software requirements are unclear and prediction is impossible within a limited period and with limited expenses, is based on the principle of finding a reasonable solution to such challenges. Thus, agile development methods have been recognized as popular and efficient approaches to resolving the aforementioned problems since 2000 [1], [2].

Agile methods are characterized by customer satisfaction, fast response to changes, and release in less time. Several agile methods have been presented that are accepted and applied to software engineering communities such as XP, Crystal Family, FDD, ASD, and SCRUM. XP was the most popular method at first, but SCRUM has become more popular because SCRUM focuses on project management via SCRUM Master [3]. SCRUM organizes the prioritized product backlogs and develops several backlogs repeatedly at each sprint period. SCRUM inherits the agile principle [4].

The key success factor of software projects is satisfaction not only of functionalities but also of quality attributes such as performance, usability, and reliability. Moreover, there are many relative points between functionality and quality attributes [5], [6], [7]. Thus, quality attributes must also be analyzed and a software development plan must be formulated that considers the relationship between functionality and the quality attributes. Indeed, when a software project must be re-designed or when it fails, the cause is not its deficient functionality but its deficient quality attribute, such as its maintenance or porting difficulty, low performance, and security problem [8]. Despite the known advantages of SCRUM, however, it has three disadvantages. First, backlogs of SCRUM focus only on functional features. Thus, it is difficult to effectively reflect the quality attributes of the software. Second, the SCRUM backlogs focus on implementing functional backlogs without a traceability analysis of the relationships of the backlogs. Thus, it is difficult to maintain software projects when their requirements are changed, because the scope of the side effects is unknown [9]. Finally, current agile methods are focused on practices that individual teams or projects need, so it is difficult to use the methods in organizations with multiple cooperating teams [10].

Many studies have been conducted to improve the problems with the agile process. Most of the studies focused on the software design approach. If there is no improvement in the software analysis approach, however, the software project would still fail due to a deficient quality attribute. Therefore, the ACRUM is proposed for the analysis and inclusion of quality attributes into software projects.

The authors contributions to this paper are summarized as follows:

- Proposal of the AQUA[2] method that analyzes the quality attributes in the SCRUM process to keep the agility of the software intact;
- Exploration of a technique to maintain the traceability

[1]ACRUM: Attribute-driven sCRUM

[2]AQUA: Analysis of QUality Attributes

978-0-7695-4490-8/11 $26.00 © 2011 IEEE
DOI 10.1109/SERA.2011.24

between a functional backlog and a quality attribute in SCRUM so as to improve maintainability via RAM[3];
- Use of the VAQ[4] technique as a validation method to judge the achievement or non-achievement of the quality attributes; and
- Objective proving of the effect of the proposed solution by applying it to a commercial project of Samsung Electronics.

The rest of this paper is organized as follows: Section 2 presents related work. Section 3 proposes ACRUM development method. The case study in Section 4 reveals that the new methods are superior to both the conventional method and the intact agile method. Finally, Section 5 presents the conclusions.

II. Related Work

In the last decade, there has been a tremendous rise in the prominence of a software engineering sub-discipline known as software architecture [11]. Architecture plays a pivotal role in allowing an organization to meet its business goals by producing a system that involves quality attributes. In other words, architectural approaches emphasize how quality attributes can be effectively embedded in software architecture [12]. In recent years, the software architecture community has developed various methods and techniques of software architecture design. For example, in [13], the author proposes a method that explicitly considers nonfunctional requirements during design. In [14], the authors propose a framework and a global analysis technique for the identification, accommodation, and description of architecturally significant factors, including quality attributes, early in the design stage.

Some quality attribute communities have developed different methods to support systematic reasoning on their respective quality attributed–for example, real time [15], reliability [16], and performance [17]–during software architecture design and review. Especially, the Software Engineering Institute (SEI) has introduced various architectural techniques to support quality attributes, including quality attribute workshops (QAWs) [5], attribute-driven design (ADD) [6], attribute-based architecture styles [18], and the architecture trade-off analysis method (ATAM) [7] / cost-benefit analysis method (CBAM) [7]. Conventional architectural methods such as those mentioned are not appropriate, however, for lightweight processes and methods. Thus, many studies are being conducted to combine the architecture-centric concept and the agile concept. Sections A and B describe a related study of two perspectives on how to combine the architecture-centric method and agile concepts.

A. Integrating Architecture-centric method and Agile

The idea of using the aforementioned techniques in an agile method could be a good solution to the problem of quality attribute non-satisfaction. In ACDM [19], [20], the problem with traditional architecture-centric methods is that in their originally authored form, they tend to be heavyweight and expensive for smaller teams and projects, short deadlines, and iterative deliveries. Overcoming these two hurdles has prevented many industry organizations from embracing these methods, and more importantly, from adopting the entire body of work (see Figure 1). In [21], the authors focus is on integrating software architecture techniques–QAW, ADD, and ATAM/CBAM–and agile development, that is, XP (see Figure 2). QAW can help a development team understand the problem by eliciting quality attribute requirements in the form of scenarios. Basing the scenarios on business goals ensures that the developers will address the right problems. In the developers work, QAW is held in the first iteration of XP and

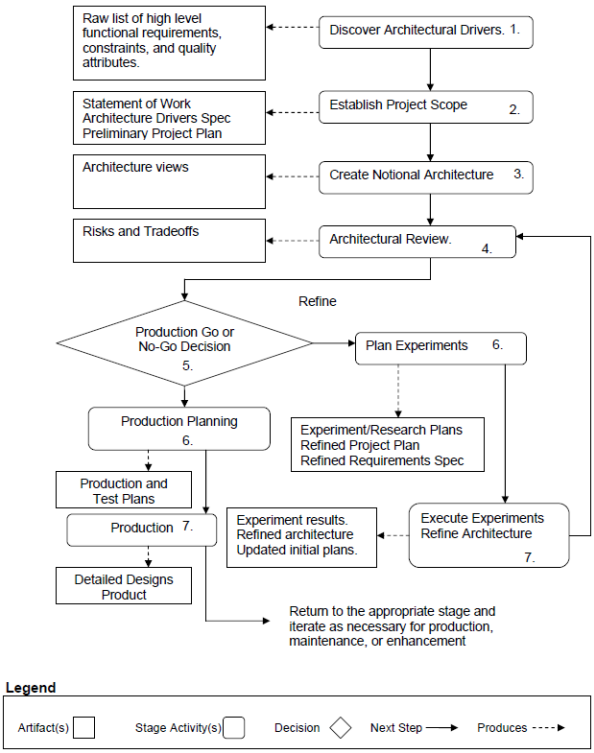

Fig. 1. ACDM Process Overview

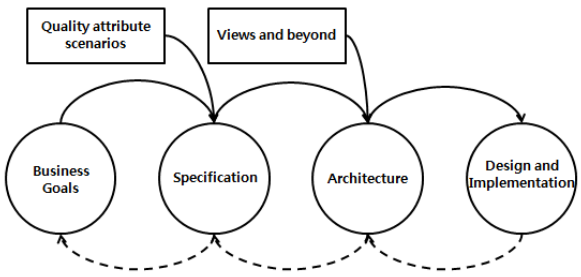

Fig. 2. Architecture-centric and XP process

[3]RAM: Relation Association Matrix
[4]VAQ: VAlidation of Quality attribute

helps enrich user stories by adding quality attribute scenarios and consequently, improving requirement elicitation.

The ADD method defines software architecture by basing the design process on the prioritized quality attribute scenarios that the software must fulfill. It helps identify the most important things to do to ensure that the design is on track to meeting key quality attribute concerns and delivering value to the customer. In this integration, the ADD method is used to design an overall architecture that is considered a basic system structure.

ATAM and CBAM provide detailed guidance on analyzing the design and getting early feedback on risks. The development team can use incremental design practices to develop a detailed design and implementation plan from the architecture. Architectural conformance and reconstruction techniques ensure consistency between the architecture and its implementation. ATAM/CBAM is an evaluation method that combines ATAM and CBAM and is used in a reported work. ATAM analyzes frequent outputs of development teams, and CBAM considers next iteration decisions based on their costs and benefits.

The aforementioned approach has some shortcomings that make it a non-practical integration. Its preparation needs more documentation, and long brainstorming sessions need to be held that seem heavy for an ordinary agile team. As discussed previously, QAW enriches user stories by adding quality attribute scenarios to them. The result of this enrichment is an increase in the amount of artifacts and the time that its session takes to be considered a sensible portion of the total project time. Moreover, ATAM/CBAM, besides its time-consuming sessions, not only creates documents about the results of its evaluation, but also needs some type of documentation as an input to start the process of its evaluation. ATAM/CBAM documents analyses, risks, decisions, and even more items as outcomes of the process, but not all of them could really be useful in an agile environment. Thus, to enrich an agile process with quality attribute concepts, these methods must be embedded into it, instead of integrating them as black boxes, Embedding means using architecture-centric methods in an agile process and conforming to the atmosphere of the agile development team and the values of the agile process.

B. Embedding Architecture-centric method into Agile

As previously stated, the idea of integrating quality attribute concepts as white boxes could be a good solution to the problem of not only quality attribute non-satisfaction but also time-consuming activities such as heavy documentation. In [22], the authors focus is on embedding the software architecture techniques of Continuous Architectural Refactoring (CAR) and Real Architecture Qualification (RAQ) into the XP development process. The CAR method helps developers reduce architectural smells by identifying smells that are discovered and solutions to revise them at the end of each task accomplishment process. Furthermore, the method was initially designed for use parallel to the XP development process so as not to affect the performance of a pure XP and to keep its agility intact. The RAQ method provides detailed guidance on evaluation quality attributes and architectural models. Such method is almost like a brainstorming session based on a working system and architectural models that embody certain steps. The RAQ method is a practice that becomes active at the end of all iterations. Its main goal is to test the working system that outputs iteration according to customer requirements. One of the most important advantages of the RAQ method is that it uses a live working system to evaluate quality attributes and architectural models.

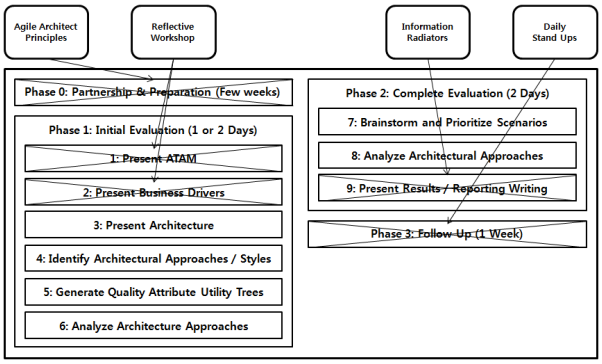

Fig. 3. Modified ATAM for Crystal Agile Model

In [23], the authors focus is modifying ATAM to map it for the crystal agile model. This mapping is very crucial to ensuring the quality of the product using the agile process model. The authors analyzed crystal and ATAM very critically and established some theoretical guidelines, on the basis of which they eliminated Phases 0 and 3 and Steps 1, 2, and 9 from ATAM to make it suitable for crystal (see Figure 3). In [24], the author proposes a method of defining how much architecture is needed for a given project, using eight dimensions (Semantics, Scope, Life cycle, Role, Documentation, Method, Value, and Cost). This is done in a workshop mode, wherein the various opposing factions have to draw a common map of the territory. These dimensions will help define a common language and a common mental model of where the various practices fit and contribute to the projects success. In [25], [26], the authors insist on the importance of balancing architecture and agility, because a well-defined and understandable architecture is needed to prevent a project from being chaotic.

Despite the advantages of embedding approach, these studies focus on software architecture design activities in the agile method without investigating how to analyze quality attributes of the software system. Thus, to enrich agile methods with quality attribute concepts, these architectural methods must not only be embedded into agile method but detailed guidance must also be provided on analyzing the quality attributes and enhancing the traceability. With these, high-quality software systems can be built.

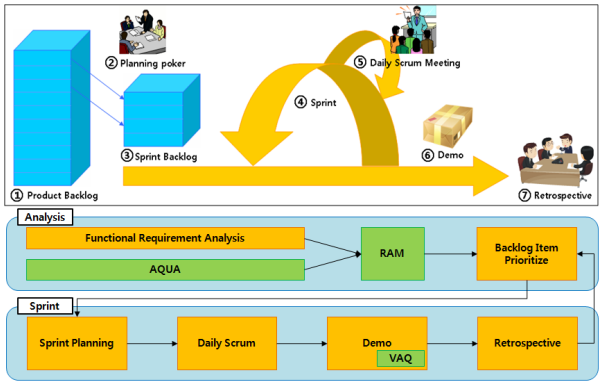

Fig. 4. ACRUM development process

III. Quality Attribute driven Agile Development Method

As previously stated, the product backlog management in SCRUM is centered on the functional requirements, without a analysis of software quality attribute. This can be connected to the failure of the software project due to an occurrence such as a late detection of quality attribute defects, a precise estimation problem on the development scope, a maintenance problem when requirements are changed, and so on. In this paper, the ACRUM is proposed as the solution to the aforementioned problem. Furthermore, the main steps in the ACRUM progress side by side with the development process of SCRUM, and do not affect the performance of pure SCRUM. Therefore, it can keep agility intact.

Figure 4 shows the development process of ACRUM. As mentioned, ACRUM was designed based on the development process of the existing SCRUM. It added three steps, however, to effectively analyze and manage quality attributes. First, the functional requirements are analyzed to comprise the product backlog (No. 1) in the first stage of SCRUM. The quality attributes are analyzed based on the functional requirements. The product backlog of SCRUM was completed by mapping out the functional requirements and the quality attributes, which are analyzed in the upper part of the figure. Then the development plan and backlog of the sprint are formulated using the planning poker (No. 2). The planning poker produces a sprint backlog (No. 3). Then the SCRUM master holds daily SCRUM meetings (No. 5) during the sprint (No. 4) progression. Finally, when the implementation of the sprint backlog was completed, the functional requirements and quality attribute requirements were verified through a demonstration (No. 6). Each sprint was composed of the activities from the sprint planning to the retrospective meeting. Such activities are repeated for each sprint.

In summary, there are three specialized activities of ACRUM. First is the analysis of quality attributes (AQUA). Second is the preparation of the correlation mapping table (RAM) between the requirements and its analysis at the upper part. Finally, the success or failure of the quality attributes is verified through the VAQ while demonstration of each final sprint. The three specialized activities go side by side with existing SCRUM practices for compatibility without extra burden. Moreover, customer collaboration, which is the core value of agile methods, can be strengthened using the method of communicating with customers through AQUA practice. RAM and VAQ practices could also further enhance the value of agile methods and speed up their response to change. In this paper, the processes of the existing SCRUM are not explained and the specialized processes of ACRUM are only explained.

A. AQUA practice

Analysis of Quality Attributes (AQUA) is a method of analyzing quality attributes using a brainstorming technique based on the business environment (see Figure 5). To enrich ACRUM with architectural concepts, AQUA method tailors existing QAW [5] as white box. AQUA practice takes place in the first stage of the SCRUM development process, after the functional requirement analysis during the making of the product backlog. The Request For Proposal (RFP) document or business context data have to be prepared before starting AQUA. The process of AQUA is as follows. AQUA originally had three phases, but ACRUM Master integrates the steps or repeats a specific step according to the situation.

Phase 0. Preparation

: In the first step, the ACRUM master introduces the AQUA process to all project staff. The ACRUM master explains the rules that all the members must follow. It also explains the objective of AQUA. ACRUM Master has to clearly explain each step so that the AQUA participant can perform

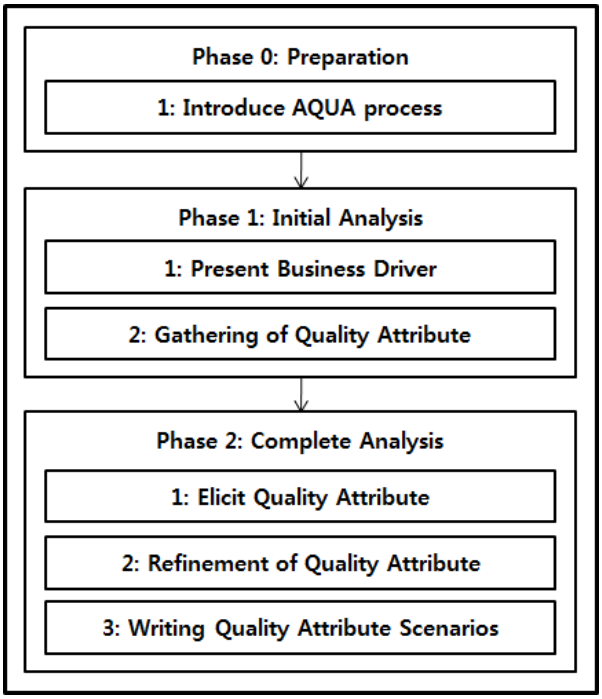

Fig. 5. AQUA process

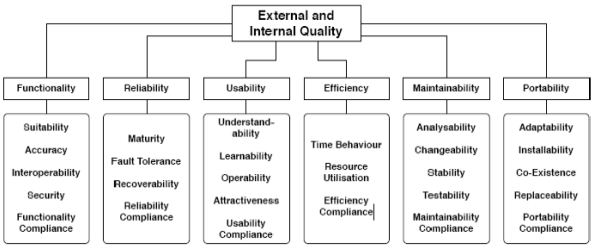

Fig. 6. ISO 9126 Model for Quality Attribute

it according to the object of AQUA.

Phase 1. Initial Analysis *:* All AQUA participants need to understand the core business driver, which means the background and motive of the software system development. That is, the business environment and the whole system from the viewpoint of its task must be understood first to be able to effectively analyze its quality attributes. For this, the customer must prepare an RFP or business context data. At this time, the following items must be explained according to priority. Although there is no document that contains all the aforementioned items, the customers RFP or business context data can substitute for what he/she or the ACRUM master explained regarding the following five items according to priority:

- The most important system function (that is major functional features);
- The technical, managerial, economic, and other constraints;
- The business goal and development background of the project;
- The major stakeholders; and
- The architecture driver (that is, the major quality attribute goals that shape the software architecture).

Other members have to elicit th related quality attributes when the ACRUM master or customer explains the five aforementioned items. At this time, they could elicit the quality attributes using ISO 9126 (see Figure 6) or the domain-specific quality attribute tree. For example, if the customer requests easy porting for many kinds of devices while explaining the business driver, other participants can elicit the portability attributes.

Phase 2. Complete Analysis *:* First, the quality attribute items that were elicited individually in Phase 1 were gathered. Duplicate items were removed or modified. Then the quality attribute items of the project were confirmed. The major quality attribute of the project was elicited and confirmed through previous activities. Such was insufficient, however, to analyze the software system. Thus, the quality attribute item had to be written down in detail using the scenario to reflect properly the quality attributes in the software architecture. For example, the quality attribute of the performance improvement of the data store latency allows establishment of the general direction of the project and knowing of what information to explain. It is difficult, however, to reflect the quality attributes in the architecture properly. In other words, they must be described in greater detail such as: Data storing or backup operation should be complete within 3 seconds in a general environment.

A scenario means a very short story on an interaction with the software system from the stakeholders point of view. Such interaction with the software system is expressed using the scenario, including the stimulus, environment, and response, to explain clearly the quality attributes to stakeholders during the analysis and to obtain enough information as the basis of the software design. The stimulation is a portion of the scenario that explains what is needed to start the interaction between the systems. The environment explains what kind of task occurs when the stimulation is generated. The peripheral condition that is needed to understand a scenario has to be explained as, for example, ”What kind of state is the software system in?”, ”What is the influence according to the abnormal state?”, ”Is the software system overloaded?”, and ”Has one of the systems processors stopped operating?”. Generally, if an environment has normal conditions, it can be omitted. Finally, the response describes how the system is operated by the stimulus. That is, it is the response to the following questions: ”Was the function performed?”, ”Is the test successful?”, ”How much effort was needed to maintain and repair the modification request?”, and others. The response has the main role of understanding what kind of quality attribute the stakeholder who is proposing the scenario is considering. The stakeholder would be concerned with the function of the system only if the response to the stimulation, such as calling for the function, is operating the function. If a stakeholder says without an error or within 2 seconds, it shows that the stakeholder is concerned with the systems reliability and performance, respectively.

In this manner, AQUA practice provides the foundation for organically managing product backlogs, including not only the functional requirements but also the quality attributes, and can lead customers to proactively participate in the project. Moreover, fields of QA backlog item are same with those of functional backlog item. Finally, it can be compatible with the intact agile process because it goes side by side with the existing SCRUM process, and only few documents must be prepared.

B. RAM practice

The Requirement Association Matrix (RAM) is the activity that maintains traceability through mapping between the functional requirements and the quality attributes. RAM practice is pursued after eliciting the quality attributes through AQUA practice and composing functional backlogs.

Quality attributes can be classified into two kinds, according to the resulting mapping type. In the first kind, the quality attribute is mapped with several specific requirements. In the

Functional Backlog \ Q.A Backlog	Data Storing or backup performance improvement	Extendability improvement of the multi format	Maintainability improvement	Security improvement of user information
Data Backup	O	X	O	X
Importing usage histories of my card	X	O	O	X
Supporting multi user	O	X	O	O
Export documentation	X	X	O	O
...	X	X	O	X
Achivement Ratio	50%	0%	100%	50%

Fig. 7. Example of RAM table

other kind, the quality attribute is not mapped with specific functional requirements, but has an effect on all the functional requirements. For example, in the case of performance improvement, it means the following two functional backlogs should be carefully implemented to reflect the performance, because the performance improvement is mapped with the data storage and report writing function. Otherwise, in the case of maintainability improvement, it is always considered in implementing a software system because it has an effect on the whole system. The quality attribute in the first case are managed by adding the quality attribute category to the related backlog item. Finally, the quality attribute in the second case is extracted by a separate backlog item, so that all sprint backlogs should always include it.

RAM Practice can be maintained traceability of a software requirements and made easier to analyze side-effects. In addition, the quality attributes are visualized by mapping with the functional requirements. Thus, RAM practice yields a more enhanced value of agile methods as responds to change.

C. VAQ practice

If the implementation of all sprint backlogs is completed in the sprint, the demonstration is performed before the retrospective meeting. The demonstration is originally for validation of the functional backlog using a working software program. VAQ practice was added to check out the achievement of the quality attribute in the sprint.

There are three validation points in VAQ practice. First, whether or not there are functional backlog items that were mapped with the quality attribute is validated. Second, the achievement or non-achievement of the quality attribute in the sprint is evaluated using a RAM table (see fig7). If all the functional backlog items that were mapped with the quality attribute have not been completed, the validation field in the quality attribute backlog item should be checked before updating achievement ratio of RAM table. The final inspection of the quality attribute will progress using the working software program after the last function is completed. At this time, if the quality attribute is not satisfied although all the functions that were mapped with the quality attribute are completed, the function should be implemented again or a new strategy must be formulated to achieve the quality attribute. Of course, the new approach must precede an analysis of the tradeoff compared with the existing backlog items.

The advantage of the VAQ method is that it prevents the problem of the projects failure due to an uncompleted quality attribute at the end of the project, by checking out the success

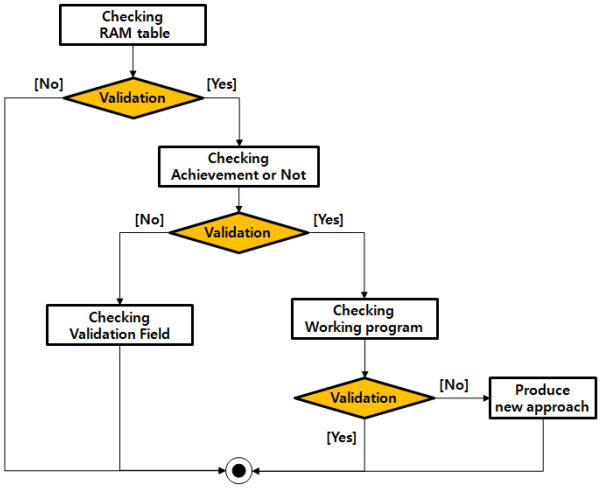

Fig. 8. VAQ process

or failure of the quality attribute in advance. It is difficult to verify objectively using only the description in the backlog item, but the quality attribute can be objectively verified with the achievement standards of the customer. Moreover, it can be more objectively verified than any other method because it uses a working software program.

IV. Case Study and Evaluation

Two evaluations were used to prove the effects of ACRUM. First, the proposed approach was compared with a related approach through the agile development evaluation check list using various parameters. Second, a quantitative analysis was performed through applying it into a commercial project of Samsung Electronics.

A. Comparison of the Two Approaches

Table I compares the proposed approach and embedding approach that were explained in Section 2. Embedding approach means using architecture-centric methods in an agile process and conforming to the atmosphere of the agile development team and the values of the agile process.

The parameters on which the comparison was based were chosen from various agilie and conventional process criteria. One of the parameters is the agile manifesto [28], the subparameters of which are the Individual and interaction, a Working Software, Customer collaboration, and Response to changes. The table also shows CMMIs factors [29], which are Requirement management, Quality assurance, Risk management, and Maintenance.

As could be understood from Table I, all approaches could improve the Individual and interaction, a Working software, Risk management, and Maintenance, but ACRUM is preferred because it considers other parameters, especially Customer collaboration, Response to changes, Requirement management, and Quality assurance.

TABLE I
COMPARISON RESULTS BETWEEN TWO APPROACHES

Parameter	Embedding approach	ACRUM
Individual and Interaction	Interaction between developers improves with various practices.	Same
Always Working Software	The software is always working through continuous integration.	Same
Customer Collaboration	Customer collaboration improves by continuously meeting customers needs.	The customer participates in AQUA and in other SCRUM steps, as before.
Responding to Changes	Response to changes improves by iterating development.	Response to changes improves with RAM. RAM makes it easy to predict side effects.
Requirement Management	Function-centric requirement management	The developers can manage the requirements, including the quality attributes, with AQUA. Moreover, they can maintain the traceability of the requirements.
Quality Assurance	Quality assurance improves through architecture refactoring and demonstration.	Quality assurance improves by checking the validation of the quality attributes and other SCRUM steps, as before.
Risk Management	Risk management improves through continuous integration and iteration of development.	Same
Maintenance	Maintenance improves through periodic refactoring.	Same

B. Case Study

The effects of ACRUM, which is the agile process emphasizing management of a quality attribute, were proven using the two principles: keeping agility and improving quality. First, it was checked if the development time was shortened to confirm the keeping agility, although three practices were added to the existing agile process. Finally, the improving quality was proved by checking the defect density and the defect fixing time, respectively.

Project Description : Both project X and Y are commercial project of Samsung Electronics those develop defense code to protect errors of NAND Flash Device. It is difficult to reuse legacy code because error correction algorithm depends on device type. Therefore, project X and Y is appropriate to prove effects of ACRUM objectively because those have little or no relationship to some experience from previous projects and emphasize quality attribute as like reliability, performance and so on.

TABLE II
OVERVIEW OF CASE STUDY

	Project X	Project Y
Applying ACRUM	No	Yes
Project M/M (Man * Month)	36 M/M	28 M/M
Defect density (Defects / KLOC)	0.45 (Defects/KLOC)	0.16 (Defects/KLOC)
Defect Fixing time (average)	3.4 days	2.5 days

The X project did not apply the ACRUM method during the project duration of January to June 2010. On the other hand, the Y project, while in the same product family as the X project, applied the ACRUM method as outlined in this paper during the project duration of September to December 2010. The details of the two project families are shown in Table II.

Keeping Agility : The most common method of examining the agility attribute is to check the development productivity. Thus, an attempt was made to check the development productivity by comparing the project development M/Ms. Project family x had 36 M/M, and project family Y had 28 M/M, as shown in Table 3. Projects X and Y are similar-scale tasks, but project Y could have improved the productivity development by eliminating unnecessary activities through applying ACRUM. For example, development duration was reduced about 22% because defect counts and defect fixing time was reduced by analysis and verification of quality attribute periodically. Therefore, the agility was maintained, although three practices were added to the existing agile process.

Improving Quality : The defect density and the defect fixing time was reduced by validating the quality attribute and easily finding the scopes of its side effects through visualization of the quality attribute. The defect densities of project families X and Y were 0.45 and 0.16, respectively. The defect density of project family Y was reduced about 64% because the quality attribute was validated periodically at each sprint. Finally, the defect fixing time of project families X and Y were 3.4 days and 2.5 days hours, respectively. The defect fixing time was reduced about 26% by intuitively finding the changing spot and easily estimating side-effect cause of defects using the requirement mapping table that can be visualized quality attribute.

V. CONCLUSION

Agile development methods are being recognized as popular and efficient approaches to development of software systems that have features such as a short delivery period, unclear

requirements, and others. They emphasize customer satisfaction, fast response to changes, and release in less time. Project failure caused by insufficient quality attributes still happens, however, because the intact agile process emphasizes functional-requirement-centric development.

To solve this problem, in this paper, three practices were introduced ACRUM method that is embedded AQUA, RAM, and VAQ practice into SCRUM to achieve quality attributes in a system. Due to the fact that these practices were derived from SCRUMs practices and values, they matched SCRUM well. The related work as like integrating approach and embeddeing approach was also discussed. The comparison with the related work showed that the proposed approach is an appropriate way to add architectural techniques to SCRUM. Finally, we have shown that development duration decreased by approximately 22%, defect density and defect fixing time decreased by approximately 64% and 26%, respectively through applying ACRUM to the commercial project of Samsung Electronics.

The limitation of this research is that only one architectural approach was suggested in the analysis phase, among many architectural approaches. If architectural approaches were applied in the design phase, the software quality would have been improved; such as with enhanced risk management and maintenance. It is not easy to apply various architectural approaches such as the agile architecture evaluation method, however, while keeping agility intact. Therefore, further research is needed to improve software quality while maintaining a balance between agility and the architectural approach.

REFERENCES

[1] P. Abrahamsson, J. Warsta, M. T. Siponen and J. Ronkainen, *New Directions on Agile Methods: A Comparative Analysis*, 25th ICSE, 2003.

[2] A. Cockburn, *Agile Software Development*, 1st Edition, Addison-Wesley Professional, 2001.

[3] Agile Survey, *http://crankypm.com/2008/10/pollresults-software-development-methodologies-agile-vs-waterfall*, 2009.

[4] Kniberg and Henrik, *Scrum and XP from the trenches*, Lightning source Inc, 2008.

[5] M.R. Barbacci, R. Ellison, A.J. Lattanze, J.A. Stafford, C.B. Weinstock and W.G. Wood, *Quality Attribute Workshops (QAWs)*, Technical Report, 3rd Edition, SEI, CMU, 2003.

[6] R. Wojcik, F. Bachmann, L. Bass, P. Clements, P. Merson, R. Nord and B. Wood, *Attribute-Driven Design (ADD)*, Technical Report, 2nd Edition, SEI, CMU, 2006.

[7] R. Nord, M. Barbacci, P. Clements, R. Kazman, L. OBrien and J. Tomayko, *Integrating the Architecture Tradeoff Analysis Method(ATAM) with the Cost Benefit Analysis Method (CBAM)*, Technical Report, 1st Edition, SEI, CMU, 2003.

[8] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 2nd Edition, Addison-Wesley, 2003.

[9] Karl E. Wiegers, *Software Requirements*, 2nd Edition, Microsoft Press, 2003.

[10] M. Lindvall, D. Muthig, A. Dagnino, C. Wallin, M. Stupperich, D. Kiefer, J. May and T. Kahkonen, *Agile software development in large organizations*, proceedings of the Agile Development Conference, 2004.

[11] C. Hofmeister, R.L. Nord, and D. Soni, *Applied Software Architecture*, Addison-Wesley, 2000.

[12] L. Chung et al., *Non-functional Requirements in Software Engineering*, Kluwer Academic Publishers, 1999.

[13] J. Bosch, *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach*, Addison-Wesley, 2000.

[14] L. Bass, M. Klein, and F. Bachmann, *Quality Attribute Design Primitives and the Attribute Driven Design Method*, Technical Report, 1st Edition, SEI, CMU, 2006.

[15] M.H. Klein et al., *A Practitioners Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems*, Kluwer Academic Publishers, 1993.

[16] M.R. Lyu, *Handbook of Software Reliability Engineering*, McGraw-Hill and IEEE CS Press, 1996.

[17] C. U. Smith and L. G. Williams, *Software Performance Engineering: A Case Study Including Performance Comparison with Design Alternatives*, IEEE Trans. Software Eng., vol. 19, no. 7, pp. 720741, 1993.

[18] M.H. Klein and R. Kazman, *Attribute-Based Architectural Styles*, Technical Report, SEI, CMU, 1999.

[19] A. J. Lattanze, *The Architecture Centric Development Method*, Technical Report, SEI, CMU, 2005.

[20] A. J. Lattanze and J. Anthony, *Architecting Software Intensive Systems A Practitioners Guide*, Boca Raton: AUERBACH, 2008.

[21] Robert L. Nord and James E. Tomayko, *Software Architecture-Centric Methods and Agile Development*, IEEE Software, 2006.

[22] A. A. Sharifloo,A. S. Saffarian and F. Shams, *Embedding Architectural Practices into Extreme Programming*, 19th Australian Conference on ASWEC, 2008.

[23] S. Farhan, H. Tauseef and M. A. Fahiem, *Adding Agility to Architecture Tradeoff Anlaysis Method for Mapping on Crystal*, World Congress on Software Engineering, 2009.

[24] P. Kruchten, *Software Architecture and Agile Software Development A Clash of Two Cultures?*, ICSE, 2010.

[25] E. Hadar and G. M. Silverman, *Agile Architecture Methodology: Long Term Strategy Interleave with Short Term Tactics*, ACM 978-1-60558-220-7, Oct/2008.

[26] D. Mancl, S. Fraser, B. Opdyke, E. Hadar and I. Hadar, *Architecture in an Agile World*, ACM 978-1-60558-768-4, Oct/2009.

[27] P. Kruchten, *The Rational Unified Process: An Introduction*, 3rd Edition, Addison-Wesley, 2004.

[28] Agile Manifesto(n.d.), *http://www.agilemanifesto.org*, Manifesto for Agile Software Development. Retrieved October20, 2008.

[29] CMMI Product Team, *CMMI for Development*, Technical Report, Version 1.2, SEI, CMU, 2006.