https://doi.org/10.48009/2_iis_2009_340-348

A HYBRID SOFTWARE DEVELOPMENT METHOD FOR LARGE-SCALE PROJECTS: RATIONAL UNIFIED PROCESS WITH SCRUM

Juyun Cho, Colorado State University-Pueblo, joey.cho@colostate-pueblo.edu

ABSTRACT

Conventional software development methods have gradually been replaced by lightweight agile software development methods since the mid-1990s. This phenomenon is mainly due to the conventional methods' shortcomings, including a slow adaptation to rapidly changing business requirements, and a tendency to be over budget and behind schedule. This paper analyzes characteristics, strengths, and weaknesses of both conventional and agile methods. This paper also explains the four major phases and nine disciplines of the Unified Process, and the common elements of the Scrum process. Finally, this paper suggests a new hybrid software development method that combines the Rational Unified Process with the Scrum process to accommodate the strengths of both methods while suppressing their weaknesses. The hybrid method can be utilized in the software industry, particularly, in the business sectors that deal with large-scale projects.

Keywords: Hybrid software development method, Unified process, Scrum process, conventional software development methods, agile software development methods.

INTRODUCTION

Conventional heavyweight, document-driven software development methods can be characterized as extensive planning, codified process, rigorous reuse, heavy documentation and big design up front [3]. The conventional methods were predominant in the software industry up until the mid 1990s. Since then, the conventional methods have been replaced by lightweight agile software development methods mostly in small-scale and relatively simple projects. This phenomenon is mainly due to the conventional methods' shortcomings, including a slow adaptation to rapidly changing business requirements, and a tendency to be over budget and behind schedule [3, 6, 9, 15, 26, 34, 36]. The conventional methods also have failed to provide dramatic improvements in productivity, reliability, and simplicity [9].

Some researchers reported that during their project development experience, requirements often changed by 25% or more [5, 18]. An interesting research mentioned that the conventional methods were not initially designed to respond to requirements change occurring in the middle of the development process, and the ability to take action appropriate to the change often determines the success or failure of a software product [37]. According to the Standish Group report [33], numerous projects with the conventional methods in various industry and government sectors were completed with fewer features and functionalities than specified in the user requirements. It is also a challenge for the conventional methods to create a complete set of requirements up front due to constant changes in the technology and business environments.

Despite the existing shortcomings, the conventional methods are still widely used in industry, particularly, for large-scale projects. The driving force of this broad utilization of the conventional methods comes from their straightforward, methodical, and structured nature [12], as well as their capability to provide predictability, stability, and high assurance [6].

Agile software development methods focus on iterative and incremental development, customer collaboration, and frequent delivery [33] through a light and fast development life cycle. There are many positive benefits of the agile approaches. Shorter development cycles, higher customer satisfaction, lower bug rates, and quicker adaptation to rapidly changing business requirements have been reported [6, 22, 24].

In spite of the potential benefits of the agile methods, many organizations are reluctant to throw their conventional methods away and jump into agile methods due to several issues. These include: 1) agile methods significantly reduce the amount of documentation and rely heavily on tacit knowledge, 2) agile methods have not been sufficiently tested for mission/safety-critical projects, 3) agile methods are not adequate for highly stable projects, 4) agile methods can be successful only with talented individuals who favor many degrees of freedom, and 5) agile methods are not appropriate for large-scale projects. Table 1 below summarizes the characteristics, strengths, and weaknesses of the conventional and agile methods.

Table 1. Comparisons between Conventional and Agile Methods

	Conventional Methods	Agile Methods
Characteristics	• Extensive planning • Codified process • Rigorous reuse • Heavy documentation • Big design up front	• Iterative and incremental development • Customer collaboration • Frequent delivery • Light and fast development cycle • Tacit knowledge within a team • Light documentation
Strengths	• Straightforward, methodical, and structured nature • Predictability, stability, and high assurance	• Short development cycle • High customer satisfaction • Low bug rate • Quick adaptation to rapidly changing business requirements
Weaknesses	• A slow adaptation to rapidly changing business requirements • A tendency to be over budget • A tendency to be behind schedule • Difficult to create a complete set of requirements up front	• Significant document reduction and heavy dependence on tacit knowledge • Not sufficiently tested for mission/safety-critical projects • Not adequate for highly stable projects • Can be successful only with talented individuals who favor many degrees of freedom • Not appropriate for large-scale projects

As shown in the table, strengths and weaknesses exist with both methods. It would be very beneficial if we can come up with a new method that accommodates the strengths while suppressing the weaknesses of both methods. This paper suggests a new method that combines the Rational Unified Process (conventional method) with the Scrum (agile method) to maximize the strengths of the two approaches.

The remainder of this paper is organized as follows: The next two sections describe the characteristics of the Rational Unified Process and the common components of the Scrum process. Next, a new hybrid model/method is presented with an explanation how the Rational Unified Process is combined with the Scrum. Finally, the author concludes the paper with a suggestion for possible future research.

RATIONAL UNIFIED PROCESS

The Unified Process (UP) is a well-defined object-oriented system development process originally offered by IBM Rational Software and was developed by Booch, Rumbaugh, and Jacobson [25]. The UP takes an iterative, requirements-driven, and architecture-centric approach, which is based on sound engineering principles [19]. Some of the universal principles of UP includes: 1) adapt the process, 2) balance stakeholder priorities, 3) collaborate across teams, 4) demonstrate value iteratively, 5) elevate the level of abstraction, and 6) focus continuously on quality.

The UP has a long and proven history and has clearly evolved [2]. Table 2 depicts how the UP has evolved through several variations. Among the many different versions, Rational Unified Process (RUP) 2003 was utilized in this study as a framework. As shown in table 2, RUP came from the Objectory process v1.0 and evolved into RUP 2003 with various additions and enhancements. More recently, a lighter UP called the Agile Unified Process (AUP) was developed for an agile software development. The AUP utilizes a streamlined approach, which has fewer activities and deliverables with a simplified method.

The structure of RUP is two dimensional: phases and disciplines. The phases represent the four major stages that a project goes through over time. The major stages include inception, elaboration, construction, and transition. The disciplines represent the logical activities that take place throughout the project.

Table 2. History of Unified Process

Year	Event
1988	Objectory v1.0 is created by Jacobson's Objectory AB company. Rational Unified Process and Enterprise Unified Process came out from the Objectory process.
1996	Rational Objectory Process (ROP) 4.0 is created. Iterative concept is introduced.
1998	ROP is renamed into Rational Unified Process and RUP 5.0 is released.
1999	Rational Unified Process 5.5 is released with an enhancement of real-time and web-based development.
2000	Rational Unified Process 2000 is developed with the addition of business engineering techniques to the business modeling discipline and a more enhanced requirements approach.
2003	Rational Unified Process 2003 is released with an enhanced test discipline.
2004	Enterprise Unified Process is developed with the expansion of the enterprise management discipline.
2005	Agile Unified Process is developed

The disciplines are divided into main disciplines and support disciplines. The main disciplines include business modeling, requirements, analysis & design, implement, testing, and deployment. The support disciplines include configuration & change management, project management, and environment. Table 3 shows the two dimensions of RUP.

Table 3. Two Dimensions of RUP

Dimensions		RUP
Phases		• Inception • Elaboration • Construction • Transition
Disciplines	Main Disciplines	• Business Modeling • Requirements • Analysis & Design • Implement • Testing • Deployment
	Support Disciplines	• Configuration & Change Management • Project Management • Environment

The nine disciplines can be employed across two or more phases during the RUP development life cycle. For example, the business modeling discipline can be utilized in both the inception and elaboration phases to understand the business environment, the requirements, the design, the implementation, and the testing disciplines can be employed across all four phases to classify requirements that the system must implement, to design a solution for the system that satisfies the requirements, to write code that makes the system actually work, and to conduct unit and integrated system testing. The deployment discipline can occur during the elaboration, construction, and transition phases to place a portion of the system or the full system into operation for users. The support disciplines can also occur across all four phases for planning and controlling the project. Table 4 shows where the nine disciplines are used within the four phases, and also where disciplines are mostly utilized. The next section describes the main objectives and activities of each RUP phase based on the explanations of Ambler (2005) and Satzinger, Jackson, and Burd (2005).

Table 4. Utilization of Disciplines in RUP phases

RUP Disciplines	Phases
Business Modeling	Inception*, Elaboration
Requirements	Inception, Elaboration*, Construction, Transition
Analysis & Design	Inception, Elaboration*, Construction, Transition
Implement	Inception, Elaboration, Construction*, Transition
Testing	Elaboration, Construction*, Transition
Deployment	Elaboration, Construction, Transition*
Configuration & Change Management	Inception, Elaboration, Construction*, Transition
Project Management	Inception, Elaboration*, Construction, Transition
Environment	Inception, Elaboration*, Construction, Transition

(* represents the most utilized phase for a certain discipline)

The Inception Phase

The primary objectives of the inception phase are to 1) identify the business scope of the new system and the project, 2) develop preliminary cost and schedule estimates based on the stakeholder concurrence, 3) identify the business need for the project, 4) understand the requirements according to the business case for the project, and 5) establish a vision for the solution. As shown in Table 4, the business modeling discipline is highly utilized in the inception phase. The main activities of the business modeling discipline in this phase include: 1) create a list of business benefits, system objectives and system capabilities, 2) describe the problem or need, 3) consider business process, workflow, and interfaces to other systems, and 4) analyze the various system stakeholders, existing system architecture, and system constraints.

The Elaboration Phase

During the elaboration phase, detailed information is gathered, hence why the requirements discipline is mostly utilized in this phase. Based on the gathered information, functional and non-functional requirements are defined. The functional requirements are activities and processes that the news system should carry out. The non-functional requirements are characteristics of the new system other than activities it must perform. Some of non-functional requirements can include technical requirements, performance requirements, usability requirements, reliability requirements, and security requirements. All defined requirements are prioritized and evaluated with actual system users. A structured walkthrough with users is an important process to make sure the gathered and prioritized requirements are correct and appropriate. User interface dialogs can also be developed in this phase.

The Construction Phase

The main focus in this phase goes to coding and testing the software. All the system components and features including user interfaces, business logics, data access functions, and help functions are implemented according to the specifications designed in the previous phase. This phase should produce a releasable working system so that the system can be deployed during the next phase. This phase can include several iterations that continue the design and implementation of the system. In particular, for large projects, several construction iterations can be involved in an effort to break the project into small and manageable tasks.

The Transition Phase

During the transition phase, the system is delivered into production and becomes available to end users. One or more iterations in this phase can involve end user training with a user's manual, beta testing to validate the system functions against end users' expectation, and corresponding modification and fine tuning. If all the requirements are satisfied, the development cycle is closed.

SCRUM PROCESS

In this section, we briefly introduce the terminologies and most common elements of the Scrum process which is one of the most broadly used agile method. The description is based on the explanation from Schwaber [28, 29, 31], who formulated the initial version of the Scrum development process. The origin of the term, *scrum,* can be found in a popular sport game, rugby, in which two teams consisting of fifteen individuals compete against each other. While the term scrum refers to the strategy used for getting an out-of-play ball back into play in rugby, it has been used to describe the process of developing products since 1986 [35]. Since then, in agile software development with Scrum, development teams are organized to have holistic move as in a rugby match, continuous interaction among team members, and unchanging core team members. Among team members, the *Scrum master* (SM) is the person who arranges the daily Scrum meeting and location, tries to remove any production impediments, and serves as a coordinator between a Scrum team and other departments. The SM also keeps track of what is going on in daily Scrum meeting to gauge the velocity of the team.

There are several important meetings in Scrum including the *daily Scrum meeting, daily the Scrum of Scrums meeting, the Sprint planning meeting*, and *the Sprint review meeting.* The daily Scrum meeting is a short (usually 5-15 minutes) standup meeting in which developers talk about what has been done since the last meeting, what will be done before the next meeting, and what the impediments are. Each development team may have different schedules for the daily Scrum meeting. The daily Scrum of Scrums is a daily meeting for Scrum masters of multiple Scrum teams. It is important to remember that both daily meetings are supposed to be a short standup meeting.

In addition to informal daily meetings, two more formal monthly meetings are usually utilized. The Sprint planning meeting is a monthly meeting where team members divide one of the items in the product backlog into a set of small and manageable tasks, which will be entered into the Sprint backlog. The product backlog contains a prioritized list of all product requirements determined by the product owner. Based on the estimated completion time for each task, the set of all tasks in the Sprint backlog is determined in such a way that they can be completed within a month. In this way, as small tasks are completed, developers can see the progress easily and can have a sense of accomplishment at every Scrum meeting. The main objective of another monthly meeting, the Sprint review meeting, is to check what has been done, what things need to be improved, and what things the team has done well. Typically, it takes a day for a Sprint review meeting followed by a Sprint planning meeting.

The success of software development with Scrum is heavily dependent on how each team is constructed. In particular, the Scrum method allows developers to run the team and have more ownership on the projects that they are working on. Developers like the idea that the team decides how to do things based on the consensus of the team, and that team has more control over how the development is managed and completed. Overall, Scrum enables the developers to have a better team work and a better communication that results in products of higher quality [16].

From the perspective of efficient product management, the product backlog and the Sprint backlog are used to help developers prioritize tasks. A Burndown chart which shows daily/monthly progress is used to keep track of task assignments. In Scrum, the priority of projects and task assignments to developers are dynamically changed based on the team members' progress on their tasks and business requirements. Another characteristic of Scrum is in its iterative and continuous monitoring of the progress of the projects. The daily Scrum meeting and the Sprint review meetings help developers monitor the progress of the projects iteratively and continuously. In particular, when there are high priority tasks, the daily Scrum meetings help developers stay on tasks and remind them what needs to be done on daily basis.

HYBRID SOFTWARE DEVELOPMENT METHOD

As described in introductory section, agile methods have mainly been utilized in relatively small-scale and simple projects and have not been sufficiently tested in the large-scale projects although researchers have reported that large-scale and complex projects also benefited from suitably tailored agile development methods [8, 10, 20, 21]. One solution to this issue is to create a new hybrid method by combining the Scrum method with the RUP method. The rationale for selecting Scrum and RUP is that Scrum is suitable for any size of projects [31] and RUP is relatively easy to streamline [2]. Figure 1 displays a suggested hybrid model.

As shown in Figure 1, the four major phases and disciplines provide the skeleton of the new method. The nine principles of RUP are reduced into seven disciplines to streamline the process. All of the seven disciplines can be utilized in each phase but only the main disciplines are displayed in Figure 1. The business modeling discipline is the main player in the inception phase. The analysis and the design disciplines are mostly utilized in the elaboration phase. The implementation and testing disciplines focus on the construction phase, whereas, the deployment and configuration disciplines are in the transition phase. Table 5 shows the seven disciplines of the hybrid model along with the original nine RUP disciplines.

Table 5. Disciplines of Hybrid Model and RUP

Hybrid	RUP
Business Modeling	Business Modeling
Analysis & Design	Requirements
Implement	Analysis & Design
Testing	Implement
Deployment	Testing
Configuration	Deployment
	Configuration & Change Management
	Project Management
	Environment

The ceremonies (Daily Scrum meeting and Sprint meeting) and roles (SM, team, product owner), and artifacts (product backlog, sprint backlog, and burndown chart) of Scrum can be embedded into a RUP phases without causing any trouble. The daily Scrum meeting, the daily Scrum of Scrums, the Sprint planning meeting, and the Sprint review meeting can be conducted iteratively in each RUP phase. The product owner can create the product log as a part of the business modeling discipline. The SM also can play the usual role defined in the Scrum process. The tasks defined in the product backlog and the Sprint backlog can be accomplished and monitored through the daily Scrum meeting and the Sprint meeting.

Figure 2 illustrates a diagram of typical phase of the hybrid model. As shown in Figure 2, the left-most column contains the product backlog and Burndown chart. These Scrum artifacts can be shared by multiple Sprints that are displayed on the top of each column. Each Sprint starts with the Sprint planning meeting and ends with the Sprint review meeting. The seven RUP disciplines can be monitored through the daily Scrum meeting. Based on the progress of a project, the usage of the seven RUP disciplines will vary. For example, in inception phase, the first Sprint may focus more on the business modeling discipline than other disciplines. However, the second and third Sprint may utilize more the analysis/design and the implementation/test discipline. In Figure 2, a typical phase consists of multiple Sprints, but each phase can contain only one or two Sprints according to the size of a project.

As shown in both Figure 1 and Figure 2, we can still provide a straightforward, methodical, and structured process in our hybrid method by keeping the four major phases in RUP. However, the hybrid method will lose some degree of predictability, stability, and high assurance because of the agility of Scrum embedded into RUP. Another gain from the hybrid method will be the capability to handle rapidly changing business requirements. It is expected that over-budget and delayed schedule issues will be reduced due to the increase of adaptability in the hybrid method. As explained, the four major phases and seven disciplines of RUP can provide a method platform, and ceremonies, roles, and artifacts of Scrum can offer management and tracking mechanisms in the new hybrid model/method.

CONCLUSION

Conventional software development methods have a potential to provide straightforward, methodical, and structured process in the software development. However, the conventional methods have shortcomings, including a slow adaptation to rapidly changing business requirements, a tendency to be over budget and behind schedule, a lack of dramatic improvements in productivity, reliability, and simplicity. Agile software development methods can provide a shorter development cycle, higher customer satisfaction, lower bug rates, and quicker adaptation to rapidly changing business requirements. Like the conventional method, the agile methods also have weaknesses, including 1) significant reduction of documentation and heavy dependency on tacit knowledge, 2) insufficient test for large-scale, mission-critical, and safety-critical projects, and 3) inadequacy for highly stable projects.

This paper presents a new hybrid model/method, which combines the Rational Unified Process with Scrum to maximize the strengths of both conventional and agile methods, while trying to suppressing the weakness of each approach. The Rational Unified Process is used as a skeleton in the hybrid method while Scrum is embedded into the Rational Unified process to offer project management and tracking mechanisms through structured ceremonies, roles, and artifacts. It would be interesting to consider integrating the principles of eXtreme Programming (XP) into the hybrid method. This might be a good future research topic.

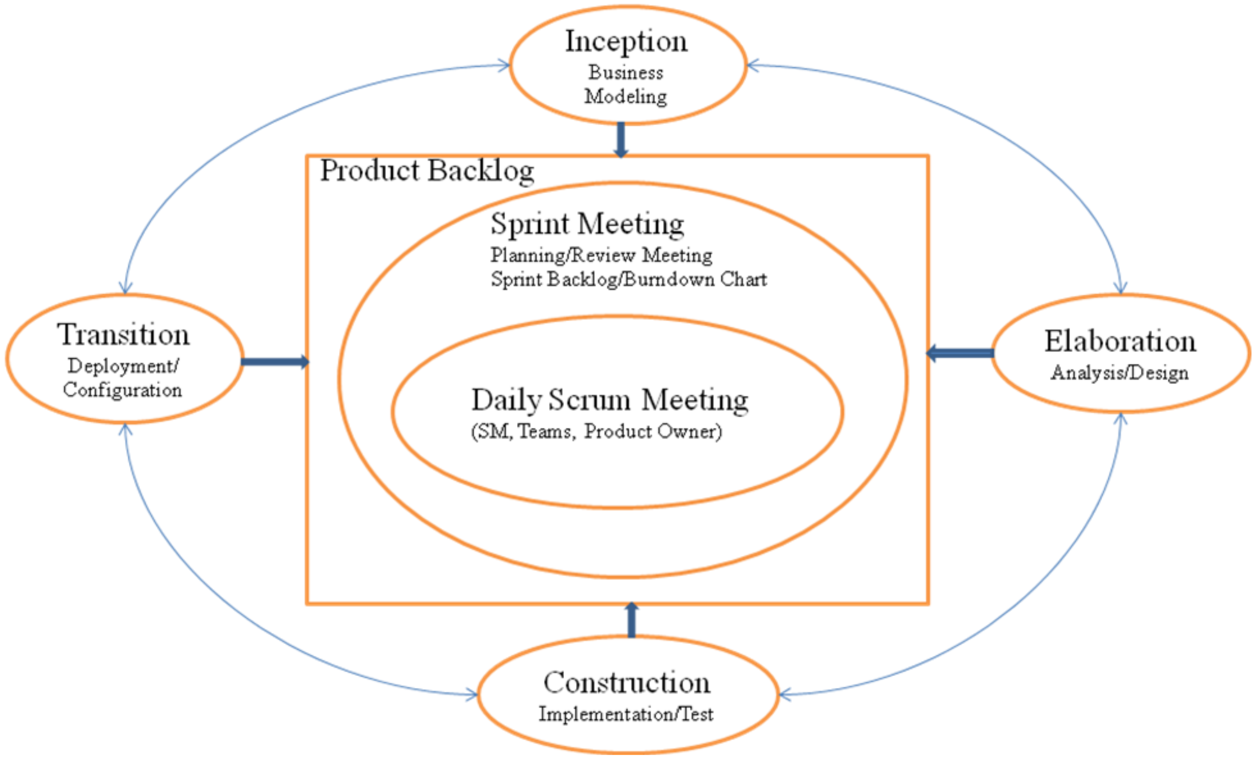

Figure 1. A Hybrid Model

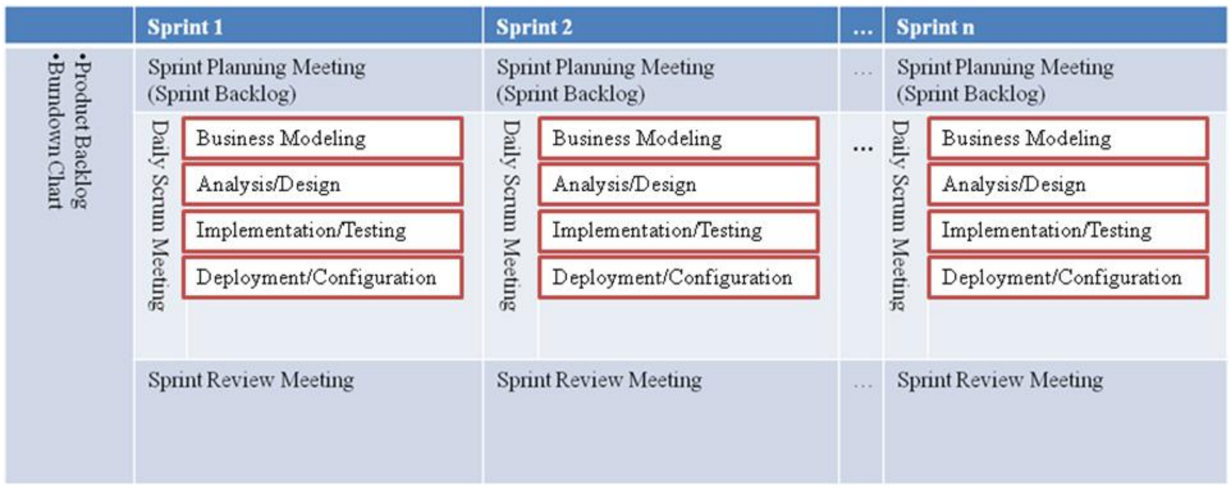

Figure 2. A Typical Phase of Hybrid Model

REFERENCES

1. Advanced Development Methods, Inc. (2007). Scrum, Retrieved March 19, 2008, from http://www.controlchaos.com.
2. Ambler, S. (2005). A manager's introduction to the Rational Unified Process (RUP). Retrieved Feb 20, 2009, from http://www.ambysoft.com/downloads/managersIntroToRUP.pdf
3. Boehm, B. (2002). Get ready for agile methods with care. *IEEE Computer*, *35*(1), 64-69.
4. Boehm, B. & Papaccio, P. (1988). Understanding and controlling software costs. *IEEE Transactions on Software Engineering*, *14*(10), 1462-1477.
5. Boehm, B., & Philip, P. (1988). Understanding and controlling software costs. *IEEE Transactions on Software Engineering, 14*(10), 1462-1477.
6. Boehm, B. & Turner, R. (2003) Using risk to balance agile and plan-driven methods. *IEEE Computer*, *36*(6), 57-66.
7. Boehm, B., & Turner, R. (2005). Management challenges to implementing agile processes in traditional development organizations. *Software, IEEE*, *22*(5), 20-29.
8. Bowers, J., May J., Melander, E., Baarman, M. & Ayoob A. (2002). Tailoring XP for large systems mission critical software development, *Proceedings of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods – XP/Agile Universe 2002*, August, 100-111.
9. Brooks, F. P. (1995). *The mythical man-month.* Reading, MA: Addison-Wesley.
10. Cao, L., Mohan, K., Xu, P. & Ramesh B. (2004). How extreme does extreme programming have to be? Adapting XP practices to large-scale projects, *Proceedings of the 37th Hawaii International Conference on System Sciences (HICSS'04),* May 15-21, Hawaii.
11. Dennis, A., Wixom, B. H., & Tegarden, D. (2005). *Systems analysis and design with UML version 2.0*. Hoboken, NJ: Wiley.
12. Fruhling, A. & Vreede, G. (2006). Field experiences with extreme programming: Developing an emergency response system. Journal of Management Information Systems, 22(4), 39-68.
13. Gall, D. M., Gall, P. J., & Borg, R. W. (2003). *Educational research: An introduction*. Boston, MA: Allyn and Bacon.
14. Glesne, C. (2006). *Becoming qualitative researchers: An introduction*. Boston, MA: Allyn and Bacon.
15. Grunbacher, P., Hailing, M., Biffl, S., Kitapci, H., & Boehm, B. (2004). Integrating collaborative processes and quality assurance techniques: Experiences from requirements negotiation. *Journal of Management Information Systems*, *20*(4), 9-29.
16. Hanakawa, N. & Okura, K. (2004). A project management support tool using communication for agile software development, *Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC'04), 316-323.*
17. Hodgetts, P. (2005). Product development with Scrum. Retrieved March 1, 2008, from http://www.agilelogic.com.
18. Jones, C. (1997). *Applied software measurements: Assuring productivity and quality*. McGraw Hill.
19. Kruchten, P. (2004). *The rational unified process: An introduction (3rd. ed.).* Reading, MA: Addison-Wesley Longman Inc.
20. Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M., Kiefer, D., May, J., & Kahkonen T. (2004). Agile software development in large organizations, *Computer*, *37*(12), 26-34.
21. Lippert, M., Becker-Pechau, P., Breitling, H., Koch, J., Kornstadt, A., Roock, S., Schmolitzky, A., Wolf, H., & Zullinghoven, H. (2003). Developing complex projects using XP with extensions, *Computer, 36*(6), 67-73.
22. Miller, K., & Larson, D. (2005, winter). Agile software development: Human values and culture. *Technology and Society Magazine, IEEE*, *24*(4), 36-42.
23. Parnas, D. (2006). Agile methods and GSD: The wrong solution to an old but real problem. *Communication of the ACM*, *49*(10), 29.
24. Parrish, A., Smith, R., Hale, D., & Hale, J. (2004). A field study of developer pairs: Productivity impacts and implications. *IEEE Software*, *21*(5), 76-79.
25. Satzinger, J. W., Jackson, R. B., & Burd, S. D. (2005). *Object-oriented analysis & design with unified process.* Boston: Thomson Course-Technology.
26. Schach, S.R. (2004). An Introduction to Object-Oriented Systems Analysis and Design with UML and the Unified Process. Boston: McGraw-Hill.

27. Schwaber, K. (1996). SCRUM development process. Proceedings of ACM SIGPLAN on Objected-Oriented Programming, Systems, Languages, & Applications (OOPSLA '96), San Jose, California.

28. Schwaber, K. (2004). *Agile project management with Scrum.* Redmond, WA: Microsoft Press.

29. Schwaber, K. (2007). What is Scrum? Retrieved March 5, 2008, from http://www.scrumalliance.org/system/resource/file/275/whatIsScrum.pdf.

30. Schwaber, K. (2007). *The enterprise and Scrum*. Redmond, WA: Microsoft Press.

31. Schwaber, K., & Beedle, M. (2002). Agile software development with Scrum, Upper Saddle River, NJ: Prentice Hall.

32. Schach, S. R. (2004). *An introduction to object-oriented systems analysis and design with UML and the unified process.* Boston: McGraw-Hill.

33. Standish Group (1994). The chaos report. Retrieved March 6, 2008, from http://www.standishgroup.com/sample_research/chaos_1994_1.php.

34. Summerville, I. (2004). *Software Engineering.* Boston: Addison-Wesley.

35. Takeuchi, H., & Nonaka, I. (1986, January-February). The new new product development game. *Harvard Business Review*, p137-146.

36. Watson, R. T., Kelly, G., Galliers, D., & Brancheau, C. (1997). Key issues in information systems management: An international perspective. *Journal of Management Information Systems, 13*(4), 91-115.

37. Williams, L. & Cockburn, A. (2003, June). Agile software development: It's about feedback and change. *IEEE Computer*, *36*(6), 39-43.