Journal of Software Engineering and Applications, 2012, 5, 367-374
http://dx.doi.org/10.4236/jsea.2012.56043 Published Online June 2012 (http://www.SciRP.org/journal/jsea)

The Success Factors of Running Scrum: A Qualitative Perspective

Rich C. Lee[1,2]

[1]System Technology Group, IBM, Chinese Taipei; [2]Department of Computer Science, National Taipei University of Technology, Chinese Taipei.
Email: rich.chih.lee@gmail.com

Received April 15th, 2012; revised May 9th, 2012; accepted May 18th, 2012

ABSTRACT

Scrum—Agile programming—is getting more attention in Software Engineering practices. Many software projects began with small and were not certain about the requirements until projects have completed; this makes Scrum more appropriate than other development methodologies. This paper reintroduced Scrum from qualitative perspective by applying ethnography and in-depth interview to two different types of project teams to articulate what the success factors are for running Scrum framework. It clearly demonstrated how qualitative research could help in disclosing the essence of facts during the Scrum adaptation in depth. It also articulated how these successful factors mutually affect to one another from System Dynamics perspective and to give further recommendations to Scrum teams and those who tend to apply Scrum development methodology.

Keywords: Software Engineering; Scrum; Qualitative Research; System Dynamics

1. Introduction

Software development is a process of communication [1]. There are many approaches [2] helping the development to produce qualified software to stakeholders. Many software projects began with small and were not certain about the requirements until projects have completed. The requirements are sensitive and volatile to environment changed. Stakeholders did not have a clear picture of what software features were in the early stage of development. Therefore Agile Programming approach was called since early-planned approaches cannot answer well to the nature of software development—the software changes and evolves all the time. Each software development project has its uniqueness and varies to others. The variance may root from the nature of the requirements, the capability of development team, the difference of development approaches, and other organizational culture-wise factors.

Scrum is an innovative approach to getting work done. Scrum is an agile framework for completing complex projects. Scrum originally was formalized for software development projects, but works well for any complex, innovative scope of work. The possibilities are endless. The Scrum framework illustrated on **Figure 1** is deceptively simple.

There are several key roles in Scrum framework: 1) Stakeholders: the mission owners, possess the idea about why to build, what to build, and how processes should be; 2) Product owner: the product development owner, closely working with stakeholders, creates a prioritized wish list-stories—called a Product Backlog; 3) Scrum Master: the facilitator, closely working with product owner, makes sure the stories in the product backlog will successfully delivered as workable sub-products; 4) Scrum Team: the developers, closely working with Scrum Master, deliver workable sub-products according to the product backlog.

Scrum Master with the team invites the product owner to debrief the stories in product backlog. Scrum Master leads a *Sprint Planning* with the team to disassemble the story into tasks, a *Sprint Backlog*. It is a Scrum team effort to decide how to implement and who should do the tasks. *Daily-Scrum* is a group meeting to discuss the progress of task implementation and technical issues. Scrum Master keeps the team focused on its goal. At the end of the sprint, the process of development, the work should be potentially shippable, as in ready to hand to stakeholders. The sprint ends with a sprint review and retrospective. As the next sprint begins, the team chooses another chunk of the product backlog and begins working again. The cycle repeats until enough items in the product backlog have been completed, the budget is depleted, or a deadline arrives. Which of these milestones marks the end of the work is entirely specific to the project. No matter which impetus stops work, Scrum ensures that the most

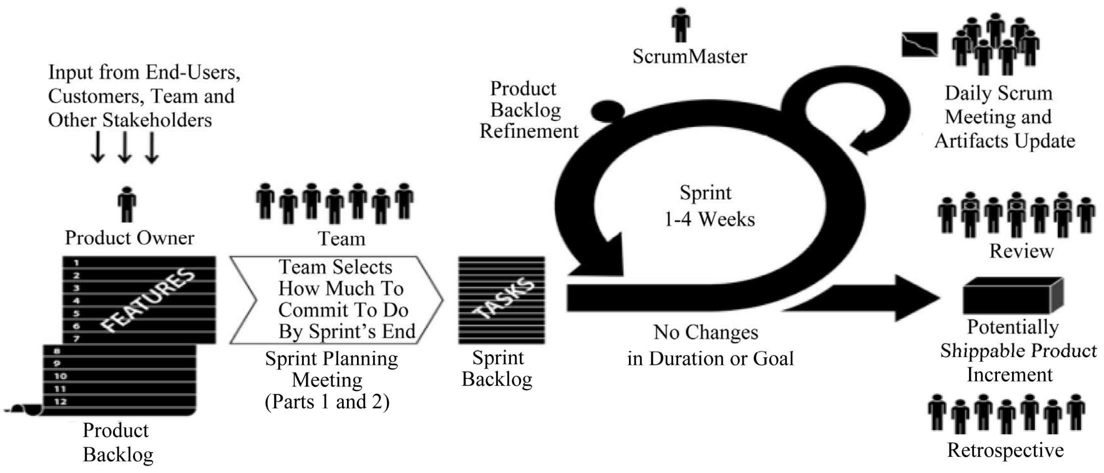
Input from End-Users, Customers, Team and Other Stakeholders
Product Owner
FEATURES
Product Backlog
Team
Team Selects How Much To Commit To Do By Sprint's End
Sprint Planning Meeting (Parts 1 and 2)
TASKS
Sprint Backlog
Product Backlog Refinement
ScrumMaster
Sprint 1-4 Weeks
No Changes in Duration or Goal
Daily Scrum Meeting and Artifacts Update
Review
Potentially Shippable Product Increment
Retrospective

Figure 1. Scrum framework. Courtesy from "the Scrum primer" [3].

valuable work has been completed when the project ends.

The success of Scrum framework requires mature capabilities such as technical and communication. Project teams are formed to achieve the strategic objectives of the organization, such as increasing market share, launching a next generation product, improving quality, enhancing customer relationships and for other purposes. Previous research on project management stressed the technical dimension of the project, such as project scheduling, resource leveling, risk mitigation and project control. However, the human side of the project, such as finding the optimal combination of team member's personality has been ignored. Unfortunately, project members play extremely important roles in achieving the project objectives and it is believed project members must be assigned based on the project needs not only technically, but also mentally. In other words, it is critical to discover if the structure of the project team can accomplish project objectives before the project begins. After adjusting the team members to meet the project requirements, project performance can be significantly improved; and if adjustment is made to the best possible balance condition, the team performance can even be upgraded to the highest level [4].

Software projects are different to one and another. It is not easy to generalize theories and to deliver tangible lesson-learn to the practitioners. The practitioners are looking for answers to improve their software development under similar culture and scenarios. To generate useful knowledge, it requires retrospection on what had not considered and what had done well from empirical cases in depth [5]. Qualitative research is a type of scientific research. In general terms, scientific research consists of an investigation that: 1) seeks answers to a question; 2) systematically uses a predefined set of procedures to answer the question; 3) collects evidence; 4) produces findings that were not determined in advance; 4) produces findings that are applicable beyond the immediate boundaries of the study.

To understand the project performance applying Scrum framework in depth, particular in resource capability arbitration among multiple Scrum teams, qualitative research can disclose the complexity of project activities and interaction among team members. This research was designed and trying to answer the following questions:

- What are the critical factors of success team communication in running Scrum?
- How Scrum Master alleviates the challenge when team members lack of skills to complete the assigned task?
- Should Scrum Masters exchange resources for specific skill required tasks?
- What are the criterions of resource exchange among Scrum teams?

2. Research Design

Qualitative research is especially effective in obtaining culturally specific information about the values, opinions, behaviors, and social contexts of particular populations. The strength of qualitative research is its ability to provide complex textual descriptions of how people experience a given research issue. It provides information about the "human" side of an issue—that is, the often contradictory behaviors, beliefs, opinions, emotions, and relationships of individuals. Qualitative methods are also effective in identifying intangible factors, such as social norms, socioeconomic status, gender roles, ethnicity, and religion, whose role in the research issue may not be readily apparent. When used along with quantitative methods, qualitative research can help us to interpret and better understand the complex reality of a given situation and the implications of quantitative data. Although findings from qualitative data can often be extended to people with characteristics similar to those in the study population, gaining a rich and complex understanding of a specific social context or phenomenon typically takes precedence over eliciting data that can be generalized to other geographical areas or populations [6].

Figure 2 illustrated the research process. This research process takes advantage of two powerful methods in deep disclosing phenomenon, qualitative research and system dynamics. In system dynamics, a causally-closed system is one in which the causes creating the behavior of interest lie within the system. A causally-closed system still is open in the sense that it can receive material, energy, random disturbances, and test inputs from outside the boundary [7]. This research relies on qualitative research to form the conceptual model—system dynamics model—for further phenomenon articulation. Most of all, by using the system dynamics tool, the researcher can give conditional parameters to see what the model behaves later to form the recommendations r policy to move the outcome towards more positive direction.

The research process begins with interesting phenomenon observed, reviewing the literature to see if previous studies could explain the phenomenon or not, a more clearer picture of research questions are formed, choosing appropriate research subjects and explain why and how the research will be conducted, undertaking a series of qualitative activities, confirming the theoretical foundation of observed artifacts via further literature review, forming theoretical models to answer the research questions, developing causal-loop analytical model, simulating various scenarios for research implication and recommendations to practitioners.

3. Data Gathering

3.1. Theoretical Background for Data Gathering

Scrum teams are self-organized, are facilitated by rich communication and a collaborative environment and are usually considered effective for colocated projects with a small team size [8]. Office layout often determines the effectiveness of team communications, it plays a powerful role in shaping a diverse range of psychological and behavioral outcomes, including individual work motivation, job satisfaction, and patterns of interactions [9]. The effective leadership of self-managing teams requires strong interpersonal and group process skills as well as external skills of information seeking and giving—plus advocacy and negotiation [10]—which is the cornerstone of makes Scrum process success. Agile Development Framework such as Scrum, requires a daily face to face communication with customers and co-workers to understand and fulfill their requirements [11]. Collaboration tools can play significant role in facilitating the team intensive communication, they are used to mitigate the communication cost. Individual leadership can help Scrum team making development decisions swiftly during the Scrum processes [12]. Although making decision swiftly does not imply the quality of outcome, but holding up issue certainly will be more devastated than need-to-readjusted decision, because Scrum process is all about embrace changes.

3.2. Data Gathering Context

Table 1 illustrated how and what information should be gathered from the informants to disclose the research questions. There were four categories of data, namely environment, communication, project, and retrospection. The major research tools were observation and interview. Collecting the data from multiple perspectives were to clarify whether these contexts having influences to the success of Scrum. The retrospection in Scrum is to create the "inspect and adapt" cycle for how a team works together and alters their practices to improve how they work by constantly asking three questions: 1) what do we need to continue doing; 2) what didn't work; 3) what do we need to start doing [13]. It is not feasible to compare the software development performance across teams. The complexity of the projects was not identical, and the cap-

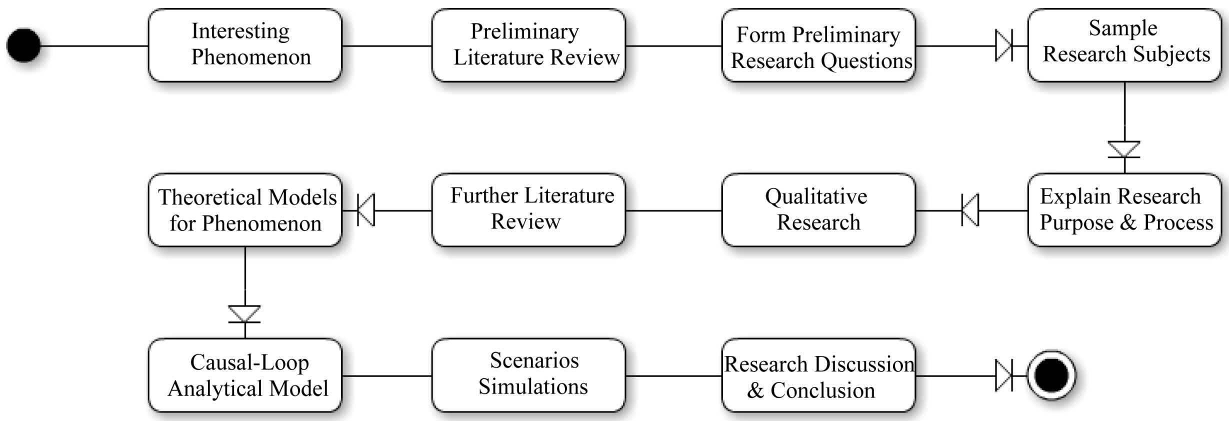
Interesting Phenomenon
Preliminary Literature Review
Form Preliminary Research Questions
Sample Research Subjects
Explain Research Purpose & Process
Qualitative Research
Further Literature Review
Theoretical Models for Phenomenon
Causal-Loop Analytical Model
Scenarios Simulations
Research Discussion & Conclusion

Figure 2. Qualitative research design.

Table 1. Data gathering context.

Source of Data	Research Tools	Context
Environment	Observation	Office Layout
		Culture and Atmosphere
Communication	Observation and Interview	Individual Leadership
		Interpersonal Skills
		Tools Used
		Vendors and External Experts
Project	Observation and Interview	Complexity
		Sprint Performance
Retrospection	Observation and Interview	Framework Enhancement
		Technological Barriers Overcome

abilities of development team were also varied. In Scrum, the burn-down chart—illustratedin **Figure 3**—is used to show, each day, a new estimate of how much work (measured in person hours) remains until all tasks are finished. Ideally, this is a downward sloping graph that is on a trajectory to reach "zero effort remaining" by the last day of the Sprint [3].

3.3. Research Subjects and Working Environment

Two Scrum teams of different companies were sampled for Qualitative Research subjects. Both teams were newbies in running Scrum. Team A is a development team in maintaining in-house developed Manufacture Execution System (MES). Team B is a small software company delivering commercial applications to various line of business.

Team A: There were several slogans placed on the walls about performance and efficiency. Team members were scattered sitting in their cubicles. The office was very quiet, people kept their voice low in conversation. They are required to use system to reserve meeting rooms before team discussion. The meeting rooms are usually fully booked. They cannot post unauthorized wallpapers in the office. Managers usually hesitate in approving posters to keep the office clean in appearance. The company does not allow employees to use instant manager tools such as MSN and Gtalk. They use emails as their major communication channels.

Team B: The office was in an over thirty-year building. Books and magazines were scattered in different places. Cafeteria area was full of foods and drinks. Project milestone charts were posted on various walls. Team members communicated thoughts via voices and with eye-contacts. The office was full of multi-channel of communication about technical discussions and requirement clarification. The team members use Internet free tools as their communication carriers. Google Chrome is

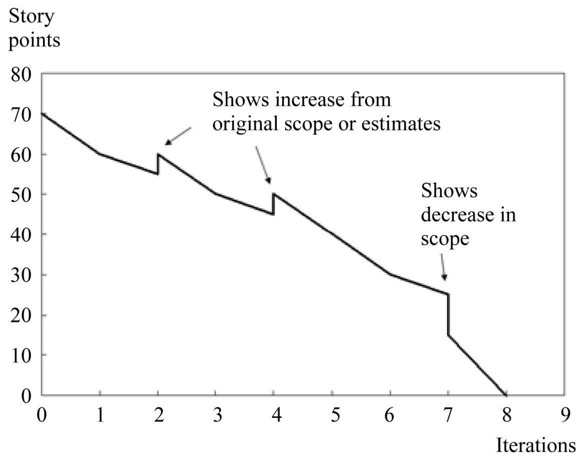

Figure 3. Burn-down chart sample. Courtesy from "crystal clear" [14].

their major workbench. There was a rather large open space in the center of the office, members frequently sat on the floor and drew ideas on large size papers. They use Group Note software as a Kanban system [15] to facilitate the software development process; the system becomes a good alternative for managers who need an easy-to-follow set of guidelines for their project management without having to resort to weeks of training [16].

3.4. Requirements and Challenges

Successfully eliciting the software requirements from stakeholders is a challenge. Requirements are considered to be a set of knowledge that facilitates business processes or help organization achieving business goals, and they are about: 1) *Customer Perspective* where the requirements regarding the quality of service desired are defined, also the knowledge concerning their perception of the service delivered; 2) *Business Perspective* where the requirements regarding the structure of the business process are defined; 3) *Employee Perspective* where the requirements concerning the skills and experiences are defined and also the knowledge used in the processing of activities based on the knowledge gathered during the previous processing steps; 4) *Product Perspective* where the requirements regarding the specification of the final product are defined [17]. Intensive communication among stakeholders, product owners, and team members are the effective approach in gathering such knowledge [18].

Team A: The software project was to migration their MES to new tentative use database. There were thousands programs in various types: 1) web applications using J2EE platform; 2) batch programs running on the database server using C++/C language; 3) database native SQL scripts. This was the first time of migrating applications from different brand of database as they used currently. They relied on external expert to guide them go through the migration process. The company wished the project could close within three months.

Team B: The software project was an inventory system running on a cloud computing environment—Google App Engine. The stakeholder wanted the system to help them managing visibility of inventory for their business partners. This was the first journey of developing applications on the platform; many required technical skillsets were not built among the team members. After serious evaluation, they chose Python as their programming language. The contract stated a penalty term if the project did not complete within six months.

3.5. Field Notes in Summary

Field Notes refer to transcribed notes or the written account derived from data collected during observations and interviews. The field notes generally consist of two parts: 1) *descriptive* in which the observer attempts to capture a word-picture of the setting, actions and conversations; 2) *reflective* in which the observer records thoughts, ideas, questions and concerns based on the observations and interviews [19]. This Participant Observation used a semi-structural field note covering important aspects of the critical successful factors of applying Scrum from literature review as follows:

1) Personal attributes of *Product Owner*, *Scrum Master*, and *Team Members* may change their reactions and decisions when team skills do not meet the project requires [20];

2) Characteristics of user stories—the requirements—are allocated to a sprint and then implemented and delivered to the stakeholders [21];

3) Project required team competence—team might not possess all skills that the project requires;

4) Team autonomy—the team had discretion, freedom, and independence in making project-related decisions, such as choosing tools/technologies, setting goals, and handling user requirement changes, and assigning personnel to the team [22];

5) Team diversity—the diversity and heterogeneity of team members' expertise areas, skills, prior work experiences, and functional backgrounds [22];

6) Team response extensiveness—how much a software team incorporates changing requirements in system scope, input data, output data, business rules/processes, data structure, and user interfaces [22];

7) Team response efficiency—the relative level of time, cost, personnel, and resources needed by the software team to respond to and incorporate a given requirement change [22];

8) Software development performance—on-time completion, on-budget completion, and software functionality are important dimensions of evaluating the software development performance [22].

Tables 2 and **3** were the field note summaries for both teams.

4. Research Propositions and Conclusion

To evaluate software development performance is complex and full of tangled factors with the characteristics of: 1) constantly changing—software requirements are volatile; 2) tightly coupled—factors are mutually affected with one another; 3) governed by feedback—the influenced factor often affects the source; 4) nonlinear—the compound behavior of a factor may have nonlinearity due to feedback loop; 5) history-dependent—factors are subject to change by their previous states [23]. Stock-Flow diagram is used to describe the complexity among these factors. The factor in box shape is called Stock which accumulates the outcome of its influenced factors. The arrow means where the influenced factor is pointed. The double-lined arrow means that the influencing factor has continuous impact on the target factor with a speed. The link means which factor is influenced. Positive sign on the link means reinforcing the impact, while negative sign means balancing the impact.

Based on the analysis from the research notes, **Figure 4** illustrated the Stock-Flow diagram of Scrum performance dynamics. The Stocks were labeled in box shapes with yellow color background. *Software Development Performance* was influenced by: 1) *On-Time Completion*—fewer schedules delayed implied greater performance; 2) *On-Budget Completion*—less budget consumed implied greater performance; 3) *Software Functionalities*—more complex functionalities implied poor performance; 4) *Development Efficiency*—more efficient in development implied greater performance. *Software Team Characteristics* was influenced by: 1) *Personal Attributes*—more willingness to collaborate with others implied positive attitude; 2) *Software Team Autonomy*—more freedom to collaborate implied positive attitude; 3) *Software Team Diversity*—more relevant skills to the project implied positive attitude. *Team Competence* was influenced by: 1) *Prior Experiences*—more relevant experiences from similar previous projects implied greater competence; 2) *External Experts*—more external helpers implied greater competence; 3) *Learning Curve*—more difficult in skillsets building implied less competence. *Software Development Agility* was influenced by: 1) *Team Attitude*—more positive in attitude implied better agility; 2) *Software Team Response Extensiveness*—more communications among the team implied better agility; 3) *Software Team Response Efficiency*—more effective communications implied better agility; 4) *Team Skills*—better team capability implied better agility; 5)

Table 2. Field note summary of team A.

Field Note Summary of Team A		
Personal Attributes	Product Owner	He has been in the company for over ten years. He often looked into the details of works and asked sharp questions. He used to argue the purpose of the software to business objectives with stakeholders.
	Scrum Master	He often goes out with members for pleasures. He usually has lunch with members if no further meeting ahead. He stressed on the details of technologies used and broke down User Stories to small tasks for members to complete.
	Team Members	They spent much time in testing software before update version. Members were very skeptical about new technologies and always asked for proven references in their sector.
Characteristics of User Stories	The User Stories were much elaborated and well planned based on the MES they had been maintained for years. Stakeholders were the key users of operations. The bottom line was to keep MES as it was before the migration. Each story was broken down to small tasks that could be completed within a short period of time, usually a day.	
Team Competence	The migration project required deep understanding about the features of new database. It also required extensive experience in migrating software over the database change. Both competences were inadequate within the team. Team spent much time in finding the appropriate solutions during the migration. They invited external experts to participate their Daily Scrum whenever it was needed.	
Team Autonomy	Scrum Master played a technical lead role rather than a facilitator in Scrum process. When there was a dispute among team members, Scrum Master made the final decision. Sometimes, team members just followed what Scrum Master implied about how to do the tasks. In the Scrum Retrospection, most discussions were laid on technical issues; as to the process improvement recommendation was rarely brought out. They needed to schedule each discussion and reserving meeting room. Many issues had been accumulated in team members' unsolved pipeline before next meeting began.	
Team Diversity	Team members have been working together for at least three years. They were trained for mastering their MES with required competences. Each member had been assigned to maintain different modules of MES before Scrum initiated.	
Team Response Extensiveness	Team members did not use any holistic software tool to manage the Scrum processes, they used spreadsheets instead. Scrum Master kept complaining about the responsiveness of updating the spreadsheets.	
Team Response Efficiency	Scrum Master asked members to bring their issues to the next Daily Scrum meeting. The meeting rooms were competing resource of the company. If the meeting room could not be booked, Scrum Master had to postpone the Daily Scrum meeting until the meeting room was available again. Team member used emails and short chats to clarify development issues with others. Keeping office quiet was a part of the company's culture and a required discipline for every employee.	
Software Project Performance	The software functionalities were clear but the team members were not familiar with the core technologies and previous experiences in new database migration. The project was delayed and exhausted extra budget for hiring external experts. External experts had no experience in running Scrum framework. They took the lead of as soon as they joined the development. They often rejected the requests of explanation why from Team members due to limitation of remaining time of the project. The project had further delayed; more budgets were consumed, because team members were reluctant to report the complexity of the system to external experts. Product Owner had disputes with the external experts about the progress of the development especially during the final phase of testing. External experts complained about the team did not report the missing links of the suggested implementation.	

Table 3. Field note summary of team B.

Field Note Summary of Team B		
Personal Attributes	Product Owner	She played a System Analyst role to the project. She was busy in eliciting the application requirements from stakeholders. She called Scrum Master several times a day asking about the progress of the project. Some stakeholders complained to General Manager of the company about her persistence in maintaining the original requirements from changing. She frequently asked team to enhance the working software to meet her standard of acceptance. Team received less requests of modifying the software from the stakeholders after her acceptance of the software.
	Scrum Master	He was a certified Scrum Master and a believer of Agile programming. He often complained Product Owner did not have enough time of explaining what requirement was in detail to the team. He had coffee meetings with the team at least twice a week. Occasionally, he invited team members to Karaoke and drinking beers together.
	Team Members	Team often went out together for pleasure during the weekends. They shared thoughts and personal experience on Facebook from time to time. Team members used Planning Poker [24] to arbitrate the argued development approaches.
Characteristics of User Stories	The project was the first time for both stakeholders and the Scrum Team taking advantage of cloud computing. Scrum Master changed the deliverables of tasks derived from stories frequently in the early phase of the project before external experts came in. Many tasks were dropped because the wrong approaches had been taken.	
Team Competence	Team members' former experiences were majorly on Microsoft technologies; few had developed application using open-source technologies before join the company. Python was chosen based on Google document suggestion. Team spent much time in learning Google Application Programming Interfaces (API).	

Continued

Team Autonomy	Scrum Master encouraged team members to have short chat whenever there was a doubt about how to proceed next. Team members posted their sample codes on the wall for technology exploration sharing. Planning Poker was not used as frequently as in their previous projects because too much uncertainty in the mastering Google APIs. Team meetings for technologic discussion were held twice a day in the early phase of development before external experts joined in the project. Scrum Master broke down the stories into tasks; some tasks were serving the same purpose because best-practice had not explored yet.
Team Diversity	Team members were hired for their expertise on Microsoft technologies. They all were familiar with database, web, and object oriented programming in using Microsoft technologies.
Team Response Extensiveness	Team members often delayed their tasks because spending too much time in discussions and answering questions from others. In order to make effective communication, they used software tools for knowledge sharing. They changed their previous behavior—ask-before-do—to check-before-ask after many best-practices had been explored. They posted important sample codes of using Google important APIs on the walls of the office. Some team members used ear sets to block the noises while working but remained alert for messages.
Team Response Efficiency	Team used Group Note software for pending issues and short conclusions of discussions. Google's Instant Messenger, Gtalk, was extensively used in communication among team members. Scrum Master asked members to clean up pending requests before off as possible; members did their best in answering the requested issues. For those unsolved requests, team members would leave them to the next Daily Scrum meeting.
Software Project Performance	The project was not severely delayed because the company had decided to hire external experts in the early phase of the development. The company allocated budget for external experts as a part of the quotation to stakeholders. The external experts had no experience in Scrum practice before. Scrum Master trained Scrum framework to external experts first before they devoted to the project. The software functionalities were delivered within budget.

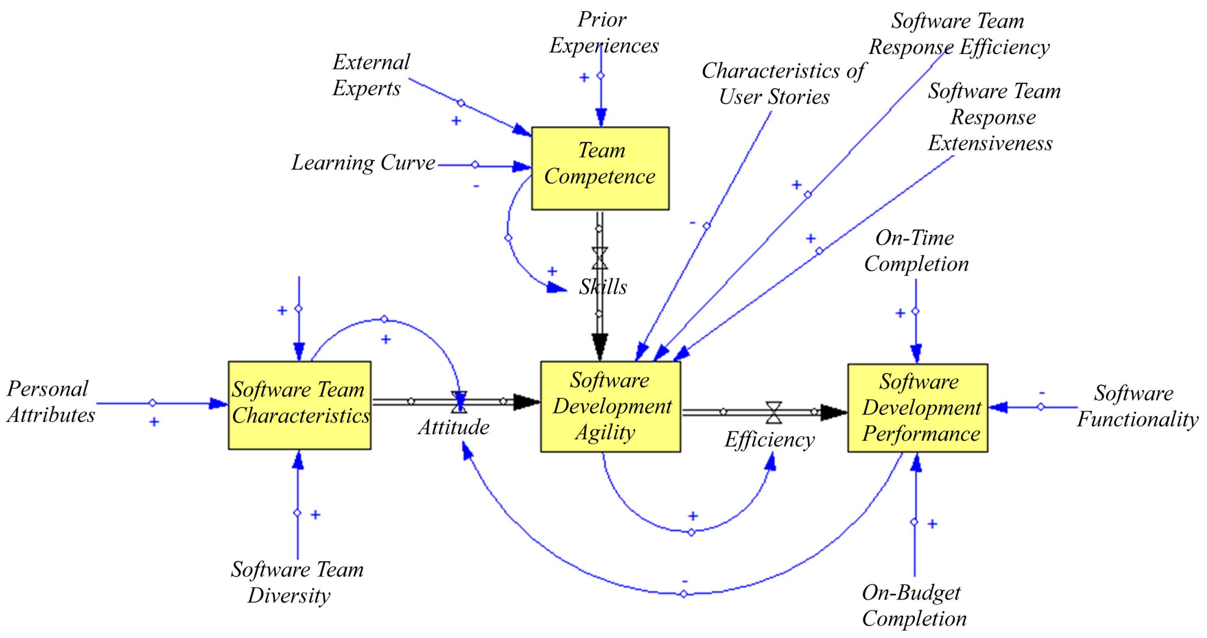

Figure 4. Scrum performance dynamics.

Characteristics of Stories—more complicated User Stories implied less agility.

Case study is a suitable research methodology for software engineering research since it studies contemporary phenomena in its natural context. This paper followed a criteria about what a good qualified qualitative research on software engineering should contain [25]. These criteria ask for the research to have clear cases with research objectives, applying theories; it must be relevant to the research questions, and making implications to the practitioners.

REFERENCES

[1] A. Gopal, T. Mukhopadhyay and M. S. Krishnan, "The Role of Software Processes and Communication in Offshore Software Development," *Communications of the ACM*, Vol. 45, No. 4, 2002, pp. 193-200. doi:10.1145/505248.506008

[2] M. Aoyama, "New Age of Software Development: How Component-Based Software Engineering Changes the Way of Software Development," 1998 *International Workshop on CBSE*, Kyoto, 25-26 April 1998, pp. 124-128.

[3] http://assets.scrumtraininginstitute.com/downloads/1/scru

mprimer121.pdf

[4] J. Y. Yeh, C. C. Wei, C. S. Wei and D. F. Lei, “The Impact of Team Personality Balance on Project Performance,” *African Journal of Business Management*, Vol. 6, No. 4, 2012, pp. 1674-1684.

[5] G. Cugola and C. Ghezzi, “Software Processes: A Retrospective and a Path to the Future,” *Software Process*: *Improvement and Practice*, Vol. 4, No. 3, 1998, pp. 101-123. doi:10.1002/(SICI)1099-1670(199809)4:3<101::AID-SPIP103>3.0.CO;2-K

[6] http://www.fhi360.org/en/RH/Pubs/booksReports/QRM_datacoll.htm

[7] J. W. Forrester, “System Dynamics, Systems Thinking, and Soft OR,” *System Dynamics Review*, Vol. 10, No. 2-3, 1994, pp. 245-256. doi:10.1002/sdr.4260100211

[8] P. Abrahamsson, O. Salo, J. Ronkainen and J. Warsta, “Agile Software Development Methods: Review and Analysis,” VTT Publications, Espoo, 2002.

[9] M. C. Davis, D. J. Leach and C. W. Clegg, “The Physical Environment of the Office: Contemporary and Emerging Issues,” *International Review of Industrial and Organizational Psychology*, Vol. 26, 2011, pp. 193-237.

[10] V. U. Druskat and J. V. Wheeler, “Managing from the Boundary: The Effective Leadership of Self-Managing Work Teams,” *Academy of Management Journal*, Vol. 46, No. 4, 2003, p. 435. doi:10.2307/30040637

[11] Y. L. Chen, “Analysis of the Agile Deployment,” Master’s Thesis, University of Gothenburg, Gothenburg, 2010.

[12] N. B. Moe, T. Dingsoyr and T. Dyba, “Overcoming Barriers to Self-Management in Software Teams,” *Software*, Vol. 26, No. 6, 2009, pp. 20-26. doi:10.1109/MS.2009.182

[13] C. Keith, “An Agile Retrospective,” *Game Developer Conference*, February 2008.

[14] A. Cockburn, “Crystal Clear: A Human-Powered Methodology for Small Teams,” Addison-Wesley, Boston, 2005.

[15] M. Ikonen, P. Kettunen, N. Oza and P. Abrahamsson, “Exploring the Sources of Waste in Kanban Software Development Projects,” 36*th Euromicro Conference on Software Engineering and Advanced Applications*, Lille, 1-3 September 2010, pp. 376-381.

[16] L. D. Rola, “Kanban for Small Software Projects,” Project Background Report, The University of Manchester, Manchester, 2011.

[17] A. E. Roger, F. N. Marcel and A. C. Lopez, “Business Process Requirement Engineering,” *International Journal on Computer Science and Engineering*, Vol. 2, No. 9, 2010, pp. 2890-2899.

[18] M. R. Haas, “Knowledge Gathering, Team Capabilities, and Project Performance in Challenging Work Environments,” *Management Science*, Vol. 52, No. 8, 2006, pp. 1170-1184. doi:10.1287/mnsc.1060.0530

[19] E. Fossey, C. Harvey, F. McDermott and L. Davidson, “Understanding and Evaluating Qualitative Research,” *Australian & New Zealand Journal of Psychiatry*, Vol. 36, No. 6, 2002, pp. 717-732. doi:10.1046/j.1440-1614.2002.01100.x

[20] M. Luz, D. Gazineu and M. Teófilo, “Challenges on Adopting Scrum for Distributed Teams in Home Office Environments,” *World Academy of Science*, *Engineering and Technology*, No. 59, 2009, pp. 308-311.

[21] J. Highsmith and A. Cockburn, “Agile Software Development: The Business of Innovation,” *Computer*, Vol. 34, No. 9, 2001, pp. 120-127. doi:10.1109/2.947100

[22] G. Lee and W. Xia, “Toward Agile: An Integrated Analysis of Quantitative and Qualitative Field Data on Software Development Agility,” *MIS Quarterly*, Vol. 34, No. 1, 2010, pp. 87-114.

[23] J. D. Sterman, “System Dynamics Modeling,” *California Management Review*, Vol. 43, No. 4, 2001, pp. 8-25.

[24] J. Grenning, “Planning Poker or How to avoid Analysis Paralysis while Release Planning,” *Hawthorn Woods*: *Renaissance Software Consulting*, Vol. 3, 2002.

[25] P. Runeson and M. Höst, “Guidelines for Conducting and Reporting Case Study Research in Software Engineering,” *Empirical Software Engineering*, Vol. 14, No. 2, 2009, pp. 131-164. doi:10.1007/s10664-008-9102-8