Situated cognition and Agile software development: A comparison of three methods

by

Khac Do Nguyen

LIU-IDA/LITH-EX-A--10/43--SE

2010-11-05
Final Thesis

Situated cognition and Agile software development: A comparison of three methods

by

Khac Do Nguyen

LIU-IDA/LITH-EX-A--10/43--SE

2010-11-05

Supervisor: Magnus Bång
Examiner: Magnus Bång
ABSTRACT

Agile programming methods have become popular in software development projects. These methods increase productivity and support teamwork processes. In this thesis, we have analyzed three well-known Agile methods - Scrum, Extreme Programming and Crystal Orange - from the perspective of situated cognition to investigate how well the methods support cognition. Specifically, we looked at how the methods aid memory and attention through the use of external representations.

The study suggests that the methods support different aspects of situated cognition reasonably well. However, among the investigated methods, Scrum stands out due to aspects of task representation (progress charts), its approaches to externalize what-to-do (memory), and the means to focus on the important programming tasks for the day (attention).

Keywords: Agile method, Scrum, Extreme Programming, Crystal, Crystal Orange, situated cognition, cognitive artifacts, representation, memory, attention.
ACKNOWLEDGEMENTS

I am heartily thankful to Mr. Magnus Bång, Researcher at Department of Computer and Information Science, Linköping University, whose encouragement, guidance and support through my thesis as an outstanding supervisor and examiner. I would like to express my sincere gratitude to Mr. Jalal Maleki, the coordinator of International M. Sc. Program, thank you for supporting me in the initial stage of my thesis enabled me to develop an understanding of the subject. Also, I would like to thank Linköping teachers and colleagues in helping me to broaden my view and knowledge. Finally, my deepest gratitude to my family, friends in supporting me.
# TABLE OF CONTENTS

1. INTRODUCTION...........................................................................................................................................3

2. BACKGROUND....................................................................................................................................................6
   2.1 Software development methods......................................................................................................................6
   2.2 Situated cognition .........................................................................................................................................15

3. METHODS..........................................................................................................................................................18

4. RESULTS...........................................................................................................................................................19
   4.1 Scrum ............................................................................................................................................................19
   4.2 Extreme Programming ................................................................................................................................22
   4.3 Crystal Orange ............................................................................................................................................24

5. DISCUSSION.......................................................................................................................................................26

6. CONCLUSION......................................................................................................................................................28

REFERENCES.........................................................................................................................................................29
TABLE OF FIGURES

Figure 1: Scrum practices ........................................................................................................... 7
Figure 2: XP practices .................................................................................................................. 11
Figure 3: Crystal Orange practices ........................................................................................... 13
Figure 4: Research process ........................................................................................................ 18
Figure 5: Example of task board ............................................................................................... 20
Figure 6: Example of burn down chart ...................................................................................... 21
Figure 7: Example of user story ................................................................................................. 22
1. INTRODUCTION

Software development methodology is a standard collection of policies, processes and procedures that is used to improve the software development process in order to increase productivity of information technology project [1]. Some of well-known traditional methodologies are Waterfall and V-Model, which introduce the sequence and structured activities. These methodologies encourage that the communication should stop when requirement specification is defined [2]. However, one of the essential characteristics of software development is that the requirements and processes can be changed during development. There are many reasons causing the requirement changes, for example misunderstanding of customer’s expression, customer cannot imagine the future system, objectives are changed, and so on.

In order to improve software development process to be adaptive to the changes and agility, as well as increasing time-to-market [3], the term of Agile Software Development Method was officially introduced in 2001 at Ski resort in the Wasatch mountains of Utah by a group of seventeen software practitioners, consultants and researchers [1]. In that meeting, the manifesto of Agile methods was introduced to focus on four values: [4]

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan
The examples of software development methodologies that align with the manifesto of Agile methods are Scrum, Extreme Programming, Crystal, Feature Driven Development, and so on. In this thesis, we would choose three Agile methods to study on, namely Scrum, Extreme Programming and Crystal; in which Scrum is the most popular Agile methods, Extreme Programming and Crystal are also well-known methods.

There are a huge number of studies and researches on how Agile methods improve development process, how these methods increase the productivity as well as the communication with customer and within development team. Many investigations on Agile methods have been conducted to understand how Agile methods support teamwork, how they increase the representation of system, the memory of team members and the attention of works. However, not many of them indicated that the representation, memory, attention are the terms of situated cognition.

Situated cognition is “the study of how human knowledge develops as a means of coordinating activity within itself”, William J. Clancey defined in the book “Situated cognition: on human knowledge and computer representations” [5]. It means that the feedback occurs internally and repeatedly over environment, which is vitally important to the development process. That aligns with the values of Agile methods.

In this paper, we will present the results of comparing Agile methods in terms of situated cognition. Although situated cognition consists of several principles, in the limited scope of this thesis, we would choose three principles to be the variables of comparison, namely representation, memory and attention. Based on these variables, three selected Agile methods including Scrum, Extreme Programming and Crystal will be compared to understand how well they support situated cognition.

**Structure**

The next chapter describes the background knowledge of three Agile methods, including Scrum, Extreme Programming, and Crystal Orange. The theories of these methods are presented with the short introduction, principles, roles, practices and artifacts. In the last part of this chapter, the theory of situated cognition is revealed, focusing on the terms of cognitive artifacts, representation, memory and attention. The third chapter explains the methodologies, which were employed to collect information and compare selected Agile methods. In Chapter four, the results of study on Scrum, Extreme Programming and Crystal Orange are presented in respect of memory, representation, and attention. In the
remainder of the thesis, we discuss on how well three different Agile methods support the terms of situated cognition. The discussion is based on the results from previous chapters.
2. BACKGROUND

This chapter describes three Agile methods: Scrum, Extreme Programming and Crystal. Moreover, theories of situated cognition in terms of cognitive artifact, memory, representation, and attention that are important in teamwork and reflected in Agile methods are also discussed.

2.1 Software development methods

**SCRUM**

Scrum is an empirical approach to control and manage complex software development, which emphasizes the ideas of flexibility, adaptability and productivity. Scrum is the most popular method among the family of Agile methods.

**Principles**

The principal fundamental of Scrum is that Scrum team organizes itself. It means project team members are not expected to follow defined tasks and plans, but organize themselves to achieve the goal of each sprint.

**Roles**

The following roles are presented according to the definition Schwaber and Beedle (2002)

The Scrum Master is responsible for ensuring that the processes of software development are taken as planned. The primary responsibility of Scrum Master is removing and changing the impediments to ensure that the team can complete their task. Scrum Master also acts as forcer of Scrum’s rules.
The Product Owner presents for customer to ensure that Scrum Team works with the right thing that customer needs. Product Owner is selected by Scrum Master, customer and management; he can be Scrum Team but cannot be Scrum Master. Product Owner writes, prioritizes and places user stories into Product Backlog, he also involves in estimating effort for Backlog items.

The Scrum Team is the project team that has the responsibility to deliver the achievement of each sprint. The team in Scrum is typically made-up of 5-9 cross-functional members (design, develop, tester and so on) who organizes and decides actions themselves.

The Customer only involves in the Scrum processes in sprint review, related to Product Backlog items.

The Management sets up organization for product development team and participates in the final decision making.

**Practices**

![Scrum practices](image)

In the end of every working day, the status meeting is conducted every day in 15 minutes limited during the sprint to keep track progress of Scrum Team by answering three questions: (Schwaber, p. 135)

- What have you done since yesterday?
- What are you planning to do today?
- Do you have any problems preventing you from accomplishing your goal?

At the beginning of sprint, the Sprint Planning Meeting is conducted, in which the Scrum Master conducts a meeting between customers, users, management, Product Owner and
Scrum Team to decide the goal and functionalities of the sprint. Then, Scrum Master and Scrum Team will meet to agree on how the goal is implemented during the sprint. The sprint planning meeting is limited to four hours only.

Sprint Review meeting is to review the works that were completed and have not yet completed. On the last day of the sprint, the Scrum Master and Scrum Team will present the results for stakeholders (customers, users, Product Owner). It will last for a time limit of four hours.

Sprint retrospective is to reflect the work in the last sprint to keep continuous process improvement. It will last for three hour limit and answers two questions: What went well in the last sprint? What can be improved in the next sprint?

**Artifacts**

Product Backlog describes everything that final product needs based on current requirements and knowledge. The Product Owner takes the responsibilities to maintain, add, remove, update, and prioritize Product Backlog items during the product development process.

Sprint Backlog lists the Product Backlog items that are implemented during the next sprint. Sprint Backlog is fixed during the sprint and when all items of Sprint Backlog are finished, a new increment of the system is delivered. The work is divided into small tasks that should require from 4 to 6 hours of work. The task board is often used to keep track the state of tasks, such as “To do”, “In progress”, “Finished”

Burn down chart gives a simple view about the sprint progress by displaying remaining work in Sprint Backlog.

**Extreme Programming**

Extreme programming is one of the most well-known forms of Agile Software Development Methods. It focuses on customers’ satisfaction and responses to the changes of requirement at any points of software development lifecycle.

Extreme programming consists of five values, they are:

- Communication
- Simplicity
- Feedback
- Courage
Respect

Keep the right communication between developers themselves as well as between developers and customers, in which face to face communication is considered as the strongest form of communication [8]. The goal is to give all the developers the shared view of system and understanding correctly the customers’ needs.

"What is the simplest thing that could possibly work?"[9]. Extreme programming encourages starting with the simplest way and does not need to predicate the future needs that may not occur.

“Optimism is an occupational hazard of programming, feedback is the treatment” [9]. Concrete feedback of current state of system leads the project toward the right direction.

The feedback can come from both developer and customers by writing unit test of pair programming or functional tests made by developers and customers.

Extreme Programming encourages refactoring the useful code and throws the obsolete code away. Extreme programming also encourages developers to solve problems with their own solutions or their colleague’s ideas.

Team members respect each other and their work, everyone in team can feel appreciated.

**Principles**
The principles are details of the practices, which suppose to make the guidance of Extreme Programming easier than practices. The followings are fundamental principles:

- Rapid feedback
- Assume simplicity
- Incremental change
- Embracing change

Feedback is vital for learning and making the changes in Extreme Programming. Customers can learn how the system gives benefit as quickly as possible in each iteration of releases. Programmer can learn quickly how to design, to implement, to test in seconds or minutes instead of days or weeks [9].

Extreme programming applies the incremental change; the prediction for the future design is a waste of efforts. XP treats every problem with the simplest solution to save for responding to changes.
Everything can be changed at anytime; instead of finding a solution for a big change; Extreme Programming separates into incremental changes.

The idea is instead of fighting against changes, developers should embrace these changes. They can put them into next iteration in spite of trying to solve them immediately.

**Roles**

There are different roles in Extreme Programming depending on specific tasks and practices. According to Beck (1999b), the roles in XP included:

The Programmer writes test before programming and keeps the code as simple as possible. The vital issue of XP is to keep a sufficient communication between developers themselves as well as between developers and other team members.

The Customer defines requirements and writes user stories. Customer is the person who decides whether each requirement is fulfilled or not.

The Tester supports customer to write functional tests, run the tests and report results.

The Tracker tracks the progress of the development and team members in each iteration.

The Coach guides and drives the team member toward the Extreme Programming processes and practices.

The Manager determines the current situations and obstacles on the progress by communicating with the project team. He is responsible for making the decisions.

**Practices**

Extreme programming is a set of 12 practices that can be described in terms of its five values, and they support each other to obtain the values of extreme programming.
The game is a meeting in each iteration of software development, that is called planning game. The customers write their requirements into user stories, the developers prioritize these stories, define the tasks and estimate time for each tasks. New requirements can be added, and old requirements can be changed or removed.

Extreme Programming encourages the short release, typically in 3-4 weeks to obtain the feedback from customers.

Everyone - customers, programmers, and manager can tell a simple shared story of how systems work using metaphor. The naming concepts should be simple to enable everyone to guess the meaning.

The design should be simple; the developer should not try to predict future needs more than the certain needs in the stories. The design must pass all unit tests.

Tests include the unit tests which developers write and the acceptance tests which customers write. As usual, the acceptance tests are automated and test system functionality.

Improve communication and simplify the code base, encourage using refactoring code continuously and explicitly.

Write software with the partner with 1 computer and 1 keyboard to find the defect as soon as possible.

Everyone takes the responsibility for the whole system, so they can change every part of the system at any time. This improves the quality of code quickly.
Build and integrate software frequently whenever the code was tested. That is Continuous integration.

Avoid the overtime to make the balance between work and life; hence the employee may have more concentration and therefore, is more interested in the work.

The customers are also the team members [10], who take the responsibilities to answer questions, resolve dispute, write acceptance tests and give the feedback of current state of system.

A convention of code has to be defined, it enables everyone to understand and change code at anytime, the coding standards are is especially important for pair programming.

**Artifacts**

Extreme Programming requires a minimal artifacts, includes story card (user story), release plan, code and unit tests, in which user stories are the most vital artifacts of XP. User stories are only simple note cards with short descriptions but it consists of requirements, predefined functionalities and negotiations among developers, customers and testers.

**Crystal Orange**

Crystal is not a specific method to manage software development but a family of methodologies that can be distinguished by their “heaviness”. In this report, we will discuss on Crystal Orange that is suitable for medium sized project ranging from 10 – 40 members.

**Principles**

Crystal contains 7 principles that guide the Crystal implementation, including:

- Frequent delivery
- Continual feedback
- Constant communication
- Safety
- Focus
- Access to users
- Automated test and integration

The results should be delivered frequently during a couple of months. In a large and critical product, the results may not be clearly deliverable but customers and product owner can look at the intermediate version to provide feedback regularly.
The Crystal team regularly meets to discuss results and project activities. The team often meets and communicates with customer to get feedback and drive project towards the true needs of customer.

Project team should be located in the same room or same facility to keep the constant communication. Moreover, all team members can access regularly to the persons who define requirements.

In most of Agile methodologies, keeping a safe zone is an essential factor, in which team members can communicate without fear of reprisal. Besides, Crystal also recognizes the safety of users to be critical projects.

Prioritize items and each team member is expected to understand two or three most important items to work on.

Crystal encourages team members to access to users of the system.

Automated test, frequently integrate and manage versioning.

**Roles**

Crystal Orange includes a number of roles, such as planning, project mentoring, architecture, technology, test teams. Moreover, it requires several new roles, including UI designer, database engineering, usage expert, technical facilitator, business analyst. (Cockburn 2002a)

**Practices**

![Crystal Orange practices](image)

*Figure 3: Crystal Orange practices*
Staging means planning for the subsequent increment of system. A product releases every 3-4 months at the maximum (Cockburn, 1998). The Crystal team selects, analyzes and prioritizes requirements that are implemented.

Each increment includes several iterations. The objective of the increment is reviewed in each iteration. That is called revision and review practice in Crystal Orange.

Tracking is to measure progress in milestones and the stability stages. Different milestones include start, review 1, review 2, test and deliver. Different stability stages include wildly fluctuating, fluctuating and stable enough to review.

Once the tracking of stability leads to “stable enough to review”, the multiple team are encouraged to achieve maximum parallelism successfully. In order to achieve parallelism, the monitoring and architecture teams review their work plans, stability and synchronization. (Cockburn 1998)

Holistic diversity is a strategy for breaking down large functional teams into cross-functional groups to enables multiple specialties in a single team. It allows forming small teams with the special know-how.

The specific methodology in each iteration is defined by methodology-tuning technique. The team can learn and use empirical knowledge for the development processes of next increment (Cockburn 2002a).

Crystal Orange encourages pre- and post-increment workshops to drive team towards development goal.

**Artifacts**

Crystal Orange requires a set of deliverables for versioning, programming, testing, communication, project tracking, and performance measurement (Cockburn 1998):

- Requirement document
- Release sequence
- Project schedule
- Status report
- User manual
- Test case
2.2 Situated cognition

In the book “Situated cognition: on human knowledge and computer representations”, William J. Clancey defined the term of situated cognition as “the study of how human knowledge develops as a means of coordinating activity within itself”. It means that the feedback-occurs internally and repeatedly over environment is vitally importance to the development process [5]. The cognition is situated; and situated cognition takes place in the real-world environment, consists of action and perception [13].

Before William J. Clancy, Greeno argued that all knowledge is bounded to social, cultural and work context [14], in which work context cannot be separated from cognition. According to David Krish [15], with the purpose of understanding the context of work, we must understand three concepts: entry points, action landscape, and coordinating mechanism. In this thesis, we only focus on coordinating mechanism and present how selected Agile methods support the coordinating mechanism.

Coordinating mechanism defined by David Krish is coordination and the mechanism that assists coordination. The coordination is the collaboration among partners associating with the resources in their environments to complete the task. The mechanism means artifacts, such as to-do list, check list, user storie which help users to manage the complexity of task [15]. The collaboration of people using the support of external artifacts develops the human’s cognition [20].

In order to understand about coordinating mechanism as well as how Agile methods support the coordinating mechanism of software projects, we will present the study of situated cognition in terms of cognitive artifact, representation, memory, and attention.

Cognitive artifacts

Norman, in his research of “Cognitive Artifacts” [16], defined cognitive artifact as “an artificial device designed to maintain, display, or operate upon information in order to serve a representational function”. He continued proving that cognitive artifacts do not only enhance human ability but also aid and improve the cognition. The roles of artifacts depend on from where and whom they are viewed. Norman pointed out that there are two different views of cognitive artifacts, including: system view and personal view [16]. In my opinion, the cognitive artifacts are the external aids that enhance human cognition and knowledge.
Some examples of cognitive artifacts are the sticky note, or task board. The sticky note enhances human’s memory by summary their memories in small note but can be retrieved very quickly when they look at the note. Task board in Scrum also a very good example of cognitive artifacts since it show the progress of working, the tasks needed to work on. It supports making a clear visual presentation and attention as well as enhances the memory of Scrum team.

**Representation**

Representation is a notion that symbolizes and explains for an entity or group of entities. In the book “Cognitive Science: An Introduction to the Study of Mind”, Jay Friedenberg and Gordon Silverman drove the ideas from Hartshorne, Weiss & Buck 1935 – 1938 and stated that there are four crucial aspects of representation: [17]

- Representation bearer – which means there must be something or someone realizing representation.
- Representation must have content – which means it must have stands for something.
- Representation must be grounded – which means it must be related to its referent.
- Representation must be interpretable by some interpreters, otherwise it stands for nothing.

Representation can be supported effectively by using external representations, for example the burn-down chart shows the progress of project, the design documents can present for the future system. Additionally, the collaboration and meeting daily of the members in team can present the progress and status of one project.

**Memory**

According to Jay Friedenberg and Gordon Silverman, “an important comment on cognitive science is that minds, unlike computers, exist in the context of a complex physical world”. Memory is the capacity of keeping information after a period of time. Memory allows us storing the past experiences to avoid and solve current problems. It affects to the capacity of learning [17]. Friedenberg and Gordon Silverman distinguished memory to three types:

- Sensory memory
- Working memory (short-term memory)
- Long-term memory
Sensory memory: is the temporary storage for incoming sensory information that is raw and unanalyzed.

Working memory: The processed information is stored briefly, but still much longer than sensory memory.

Long-term memory: The information is stored over prolonged time and can be retrieved to use in the future.

Attention
Attention is the cognitive process of selecting specific goal and concentrating on these goals while ignoring other things. Selected attention chooses an element from around environment or from long-term memory to map it to working memory. All mental actions require limited cognitive resources at a moment, so that we can not do all at once time. Therefore, we must drive our attention within a limited intentional budget. One of the most famous quotes about attention is from William James from 1890 "Everyone knows what attention is. It is the taking possession by the mind, in clear and vivid form, of one out of what seem several simultaneously possible objects or trains of thought." [19]

In order to increase attention, the projects in many fields often divides the tasks to the small tasks and each members should only understand and focus on their task until they complete their tasks.
3. METHODS

During the thesis work, the qualitative method was conducted to collect the literature study on Agile methods, especially in Scrum, Extreme Programming and Crystal. After researching on Agile methods, particularly in processes, principles, roles and practices, the literature data of cognition and teamwork was elicited. Then, the ideas of this thesis were narrowed on comparing Agile methods in terms of situated cognition.

Among the aspects of situated cognition, we selected four aspects to focus on this thesis, they are: Cognitive Artifacts, Representation, Memory, and Attention that are important in improvement of software development processes. They are the variables of the comparison, together with tools that support them. Based on the empirical case studies in articles, we carry out the comparison of Scrum, Extreme Programming and Crystal in terms of Representation, Memory, and Attention. The figure 4 shows the research process.

![Figure 4: Research process](image-url)
4. RESULTS

This chapter describes the results of study on three selected Agile methods in terms of situated cognition. The results are presented by researching through Scrum, Extreme Programming and Crystal Orange, focusing on their processes, role and practices, in order to see how well they support memory, representation, and attention of situated cognition theory. The next chapter discusses and presents the comparison among Scrum, Extreme Programming and Crystal Orange based on these results.

4.1 Scrum

Cognitive artifacts of Scrum

Scrum relies heavily on different external tools to support the programming process. These tools are cognitive artifacts since they support memory and direct attention to the important tasks. For example the task board and burn down chart give very quickly the progress of sprint and what has to be done. Without remembering in the mind, Scrum team can gain very quickly memories in task board that lists sprint backlog items, works to do, works in process, works to verify and completed works. The Scrum team does not need to remember lots of things but retrieves the memories very quickly in the beginning of working day and releases them in the end of working day. It also drives the attention of team member toward their tasks only and can see the representation of tasks and their progress. The following picture is an example of how task board looks like.
The other aids and practices that support situated cognition will be presented more in next parts of representation and attention.

**Representation**

Product backlog and sprint backlog lists play the vital roles in Scrum team as artifacts that enhance the representation of works and products in Scrum project. Product backlog contains the current requirements that present for customer’s need as well as for the system that will be developed. The structure of the logs, that is structured list, is physical representation of the most important tasks and this clearly aids cognition and attention. However, requirements are also the center of all other Software Development Methods. The difference between Scrum and other methods is that the product backlog is regularly updated to encourage customer’s feedback. Therefore, the representation is closer to the thing it should be. Multiple actors involve in defining product backlog items, such as customer, project team, user, management. At every sprint, the product backlog items are reviewed by Scrum team.

In Scrum, the burn down chart is used in each sprint to give a simple view of sprint’s progress. It totally supports for the representation of Scrum team’s progress.
The purpose of daily Scrum meeting is not to show the status of works, but for each team member to explain what they have done, what they are going to do and what obstacles are on their work. These practices support the representation of members’ works and their performance in a certain day, which is extremely hard to understand clearly in other methods.

In sprint review meeting, Scrum team will present the results of the last sprint for stakeholders that gives a plain presentation of what has been done and what the system will look like when project finishes.

**Attention support**

Scrum processes and practices support attention-demanding processes since the clear goals of product are selected into product backlog lists with the agreement among all actors in Scrum project. Product backlog defines everything that is needed in the final product and it will be maintained regularly based on the changes of requirements and environment. However, product backlog seems to be too general for supporting attention. Therefore, Scrum introduces Sprint backlog that contains selected product backlog items to be implemented in the next sprint. These items are selected by Scrum master and Product owner during sprint planning meeting, but they need to be agreed upon Scrum team to make it reliable. Then, the specific tasks will be assigned to each members of Scrum team. These practices narrow down the attention of each team member by only focusing on their tasks. In sprint backlog, anything that is not to be released in this delivery will not be included, that would increase the attention of Scrum team.
Daily Scrum meeting intends to review the tasks of each member in the day that could force members to concentrate on their task in one working day. In addition, daily sprint meeting also drives Scrum team’s attention toward specific tasks in the next day. Finally, one more purpose of daily sprint meeting is to discover and remove impediments to tasks completion, which means team members can solve these impediments in the next days or can ignore it to concentrate on their assignments. It encourages the term of attention in situated cognition since the Scrum team members concentrate to complete their assigned tasks, not on others thing, such as solving impediments.

4.2 Extreme Programming

Cognitive artifacts of Extreme Programming

Extreme Programming introduces several artifacts to enhance situated cognition of team members. Extreme Programming encourage that everything should be treated in the simplest way, therefore it promote the simple design, simple code that can be considered almost as cognitive artifacts to support memory and sharing knowledge. Similarly to product backlog in Scrum, Extreme Programming includes lists of user story cards that describe customer’s requirements. These cards are only small sticky notes to enhance the memory and attention. When the team decides to work on specific requirements, they will ask customer to get the details of requirements for these tasks.

![Image of a sticky note with a user story]

**Figure 7: Example of user story**

In Extreme Programming, metaphor means that everyone including customers, developers, and manager can tell a simple shared story of how systems work. The naming concepts should keep as simple as possible, so that everyone can guess the name. This practice and
representation, together with simple design, does not only aid cognition but also encourages the memory of reader since XP team can easily recall and understand topics when reading design and coding documents.

**Presentation**

Extreme Programming provides lists of user story cards that describe customer’s requirements. These cards represent for the features that the system will contain. At the beginning of the project, a prototype of system is built to test current technology and architecture possibilities. This prototype can be seen as a cognitive artifact that contains and represents a “memory” of how the future system should be (i.e., embodying features and requirements). The significant differences between user story cards and requirement documents of other methods is that user stories only provide brief but enough information to estimate effort to implement. When it is implemented during project, the story developer will meet the customer to get details of these stories. In this case, the card work memories and these small cues aid the discussion with the customers and drive the discussion.

Extreme Programming encourages Test-Driven Development, meaning that the unit tests cases are written and run before the actual programming. Customers are also involved in writing functional tests (acceptance tests). These practices support reflection-process that aid programmers to understand the problem and create “mental” solutions even before the system is built.

During the development, customers are required to involve in Extreme Programming team. They do not only give quick feedback about the development, but also represent for the true need that product will contain in the end of project. The close collaboration with customers aid programmers because all people that are involved in the project memorize different things; hence, these practices can be seen to create a distributed memory of work-to-do in the project (that is a cognitive practice).

**Attention support**

Just like most Agile methods, the short release practice of Extreme Programming enables the concentration of XP teams on a few important requirements. The customers involve during project and they are the people who select stories for each iteration. This practice drives XP team toward the features that customers need the most, hence XP seems to be supporting attention in this regard.
Test-Driven-Development means that test is written before coding. It drives developer’s attention to implement right codes, avoid the case of writing test cases to be adapted to written codes. That would be drive XP team toward the right customer’s need.

Extreme Programming also introduces daily stand up meeting as Scrum, but this meeting is conducted in the morning to answer three questions similarly to Scrum: What have you done since yesterday? What are you going to do today? What are obstacles on your way? The daily meeting does not only support representation and memory, but also supports attention of cognition by driving XP team to focus on the tasks that need to be completed.

4.3 Crystal Orange

Cognitive artifacts
Crystal Orange introduces a set of deliverables for versioning, programming, testing, communication, project tracking, and performance measurement (Cockburn 1998) that can support situated cognition since they enhance memory, representation and drive attention focusing on a few tasks at one time. For example, Requirement document, release sequence, project schedule, status report, user manual.

Crystal Orange also introduces short iteration, normally in three or four months that are longer than in Scrum and Extreme Programming. The team participates in selecting requirements and estimating efforts to complete their tasks as well as the next increment. This practice together with its artifacts encourages cognition, in term of memory compared to other traditional software development methods.

One of the different characteristics of Crystal Orange from the other Agile methods is the methodology-turning technique. In every iteration, Crystal team learns and uses achieved knowledge to define adequate development processes for next iteration. It can be considered as a cognitive practice in which the memory can be kept in long-term and retrieved to solve the future works.

Representation
Crystal Orange requires a set of deliverables, including requirement document, release sequence, project schedule, status report, user manual, test case, and so on. Although they are not really light-weight compared to Scrum and Extreme Programming, they still encourage the representation of cognition by presenting the tasks in the reports.
Each iteration of Crystal Orange project consists of following activities: construction, demonstration, and review the goals of this increment (Cockburn 1998). These practices give a clear representation of iteration’s objectives. Therefore, it would reflect the representation aspects of situated cognition theory.

The progress in Crystal project is measured and tracked by milestones and stability stages. In terms of representation, we would consider that this practice plays the same role with the burn down chart in Scrum that is a cognitive artifact.

**Attention support**

In the same way with most Agile methods, the project and requirements in Crystal Orange are divided into small stages. The most important requirements will be solved first and that can increase the attention of Crystal team. Each of team members is expected to know and work on two or three most prioritized requirements. Crystal also introduces the review practices to examine the objectives of the increment, so the goals and attentions are increased to the maximum. Crystal Orange also encourages the safety of developers to complete their tasks and removes the obstacles in their way. That would aid the attention of the members in Crystal team.
This thesis investigated how well modern software development methods support situated cognition. Scrum, Extreme Programming and Crystal divide project into small releases, and each team member only need to remember a few tasks in each iteration. This practice can reduce the workload of memory since team members do not need to remember large number of tasks. Therefore, it supports the memory of all people involving in the software projects. Scrum and Crystal Orange encourage the review in the end of a sprint or at the beginning of a stage, which would retrieve the memory of last iteration to gain a better result in next increment. In Extreme Programming, the refactoring process also encourages reuse of code. These practices could support memory and understanding of the problems since the codes are familiar.

Both Extreme Programming and Scrum encourage the use of sticky notes to describe user requirements and programming tasks. They can be considered as cognitive artifacts that enhance memory and support sharing of knowledge within programming team. Especially, the burn down chart and task board in Scrum - as presented in the previous chapter support very well memory by showing progress as well as programming tasks. In the aspect of memory, these Agile methods support quite well aspects of situated cognition. However, we rate Scrum as the best method in this regard.

Scrum, Extreme Programming and Crystal Orange have different ways to present things, for example, the product backlog in Scrum, user story cards in XP as well as the deliverable sets in Crystal Orange represent the customer’s need and product’s
requirements. They are reviewed and updated regularly and can be seen as external representations and embodiment of the user’s requirements.

Both Scrum and Extreme Programming introduce special practices such as daily meetings where the programmers report what they have done and will do tomorrow. This can be seen as a cognitive practice to remember the important tasks and keep track of the project. This practice is not included in Crystal Orange.

Extreme Programming encourages test-driven-development, which means writing and running tests before the actual coding starts. It does not only drive developers toward better programming constructs but it also makes a representation of the system based on the requirements before the system is formed. Scrum does not contain this practice, but the task board and burn down chart in Scrum make a clearer representation of the programming progress than Extreme Programming does. To conclude, from the perspective of situated cognition, it seems that Scrum and Extreme Programming support aspects of memory and attention in the software project more appropriately than Crystal Orange.
6. CONCLUSION

In this thesis, we have investigated how well modern agile programming methods support situated cognition. That is, how they aid programmers remembering tasks and how they structure project-specific problems. The study investigated three methods: Scrum, Extreme Programming and Crystal Orange. The results indicate that all methods support situated cognition aspects quite well due to the introduced artifacts and practices. Among the methods, Scrum seems to support situated cognition particularly well since the method introduce several cognitive artifacts that aid everyday programming. These artifacts structure activities in ways that offload cognitively-demanding tasks and direct attention to prioritized work-to-do by using special external representations. Moreover, the everyday meetings of Scrum support sharing of knowledge and team awareness of the tasks ahead.
REFERENCES


8. Chris Loftus, Mark Ratcliffe. Extreme Programming Promotes Extreme Learning? University of Wales, Aberystwyth, Department of Computer Science


10. James Newkirk. Introduction to Agile Processes and Extreme Programming. ThoughtWorks, Chicago, IL, USA


18. Picture of Task Board.
   http://www.mountaingoatsoftware.com/system/hidden_asset/file/33/LabelledTaskBoard.jpg (2010/10/04)


The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© [Khac Do Nguyen]