Comparison of Software Engineering Processes

Cheuk Hei CHAN
Kimberley Ling Zhen CHONG

Table of Contents

Introduction

In this paper, we will first define agile processes and investigate the different types of techniques used in agile processes. We also present some of the tools and practices that can be used in pair and team working. Next, we identify processes, techniques and tools as well as assessing risk factors in the second coursework. We then reflect this pair working experience as a conclusion to the coursework.

Agile Processes

Identification of agile processes

According to Agile Alliance (2015), agile is defined as the "ability to create and respond to change in order to succeed in an uncertain and turbulent environment". Agile processes are used as project or software development processes that are based on iterative development (Mohammad, Alwada'n, & Ababneh, 2013). They pose as a model of a software development life cycle that is used to develop a software project in either a short or long run. Based on the Manifesto for Agile Software Development (Beck et al., 2001), there are four core values and 12 principles that agile development processes would follow. The manifesto stated the four core values as: individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan.These values and principles forms the foundation of agile processes stemmed from traditional approaches such as the waterfall and spiral models.

Description of agile processes

According to Layton (2012, pp. 63), agile processes are comprised of iterative development measured by working software and are made up of self-organized and multifunctional teams. As agile processes are iterative, development can be done in parallel in a team by incrementation instead of building a programme all at once (Snyxius, 2016). Layton (2012, pp. 63) also mentioned that agile processes emphasizes on "simplicity, transparency, and situation-specific strategies". Compared to traditional development methods such as the waterfall approach that is done sequentially, agile methodologies are adaptable to change and progresses rapidly. In this paper, the existing agile processes that will be mentioned are: Scrum, Crystal, Dynamic Systems Development Method (DSDM), Feature-driven development (FDD), Lean software development (LSD), and Extreme Programming (XP).

Classification of agile processes

Iacovelli and Souveyet (2008, pp.98) regrouped major components in some of the agile methods and classified them into three classes: Software Development Practices Oriented Methods, Project Management Oriented Methods, and Hybrid Methods (Figure 1).

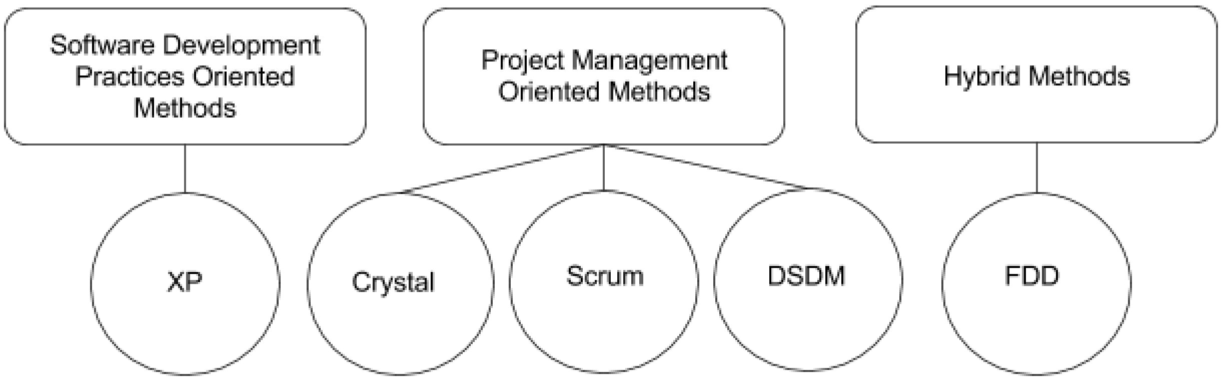
Software Development Practices Oriented Methods
Project Management Oriented Methods
Hybrid Methods
XP
Crystal
Scrum
DSDM
FDD

Figure 1. Classification of agile processes into different types of methodologies

In software development practices oriented methods, the processes are characterised by their short iterations, small scope of projects, and small sized teams. These processes have specifications on the development and focuses on increasing the integration, precision, quality and productivity of the product (Iacovelli and Souveyet, 2008). They further described project management oriented methods as processes suitable for large sized complex projects. Meanwhile, hybrid methods inherits the development practices from software development practices oriented methods as well as guidelines from project management oriented methods. This classification is criticised by Schuh et. al (2016) for being incomplete and hinders use for complex projects. Nevertheless, the classification still contributes in the selection of appropriate processes in a development life cycle and a potential to improve agile processes. Although Iacovelli and Souveyet did not mention LSD in their article, LSD was founded by Toyota that was based on efficient manufacturing by inexperienced workers to complete complex processes in their factories (Layton, 2012). Based on the scale of it, it would be presumed that LSD either falls in project management oriented methods or hybrid methods.

Major Techniques Used In Agile Processes

Scrum

Scrum is one of the frameworks used in agile processes which designs for development teams. It is a management oriented methodology that mainly focus on organizing the project. There are three main parts in Scrum which is transparency, inspection and adaptation. Transparency means that anyone who responsible for the project must have visibility on the process. The team also need to frequently inspect and track the progress of the work. For adaptation, it means that the team should adjust the process to prevent from deviation of the goal (Schwaber & Sutherland, 2013).

There are three main roles: product owner, scrum master and scrum team. Product owner communicates with the customers and is responsible for satisfying their requirements and expectations of the product. Scrum master works with the scrum team to ensure work flow is smooth and tasks are completed on time. Scrum team works together to create the final product.

Product backlog is a list of tasks that need to be completed. Sprint refers to the iteration of a scrum. For each sprint, items in product backlog are selected to be sprint backlog. Scrum team accomplishes tasks in the sprint backlog and increment them into the final product through sprint iteration.

Techniques	Description
Planning Poker	It is used to estimate relative size of items in product backlog. Firstly, briefly introduce and discuss each item. Then, participants select card and face down to represent the estimation of the item. Everyone turn the cards over at the same time. People need to explain their estimations if they are in high or low estimates. Participants discuss and estimate the item until consensus is made. This technique can prevent influence from other team members. So, they can think independently for the estimations.
Sprint Planning	Define sprint goal that should be finished at the end of the sprint. Then select items that the team want to completed in the sprint from product backlog. After that, the selected items are divided into tasks put them in sprint backlog.
User Stories	It is a description of requirements from customers in a storytelling mode. It is short, simple and easy to read. Template: As a <role>, I want <goal> so that <reason>. (Berteig, 2014)
Sprint Review	It holds at the end of each sprint. Presenting completed works to stakeholders, decide what is the next steps.
Sprint Retrospective	At the end of a sprint, the team reflects works in the past sprints. Think of solutions to solve the existing problems and tackle them in the future sprints.
Daily Scrum	Daily 15 minutes meeting which reviews the progress of the project. Participants need to answer three questions: "What did I do yesterday?", "What do I need to do today?", "Is there any obstacles stop me from completing my work?".

Table 1. Major techniques used in Scrum (Measey & Radtec, 2015, pp.131-140; Williams, 2012, pp.74-76)

Crystal
Crystal methodology, also known as Crystal Family is developed by Alistair Cockburn. It is adaptive which can adapt to XP or Scrum techniques with no limitation on any tools or development practices. It is also lightweight that does not requires much documentation and intermediate work products. It emphasizes on the communication and interaction of people but not the process itself. (Cockburn, 2000, pp.163-167)

Crystal suggests the use of colours to represent the size of a project such as Crystal Clear (the lightest), Crystal Yellow, Crystal Red, Crystal Sapphire (the darkest). When the color is darker, the size of the project is larger. It also categorizes projects into four critical levels: Comfort (C), Discretionary Money (D), Essential Money (E), and Life (L). i.e. If there is a probability to cause mortality, then that plan would be in the L level. (Coffin & Lane, 2006)

Techniques	Description
Blitz Planning	It provides a great opportunity for executive sponsor, developers and users working together to create a project map and timeline. First, the team write all the tasks into cards. The dependency, priority, person's name and working time are added onto the cards and put in sequence. If a person is responsible for too

	many tasks, that person is off-loaded.
Reflection Workshop	After each delivery, the development team holds a reflection workshop for an hour. The team members discusses what good practices they should keep, what problems are they facing and what improvements to try in the next period.
Daily Stand-Up Meetings	It is a short meeting for team members to exchange notes for identifying problems. They need to stand up during the meeting to prevent them from falling asleep or any distractions.
Side-by-Side Programming	It involves two people sit next to each other, but work on their own tasks. So that they can examine other's code, test it and comment on it quickly.

Table 2. Major techniques used in Crystal (Cockburn, 2004, pp.79-127)

Extreme Programming (XP)

Extreme Programming is a lightweight and software development practices oriented methodology for development teams of any size. The team just needs to satisfy the customers and adapt to any changes to the requirements from them. It emphasizes teamwork, communication and programming techniques. There are some characteristics which is different from other methodology. For example, XP uses short and frequent development cycles to create early feedback. It relies on automated tests, close collaboration and oral communication to increase the quality of the products. It also adapt to any change from the customers with flexible planning. (Beck, 2005, pp. 1-7)

Techniques	Description
The Planning Game	The aim is to steer the project into delivery. Customers write their requirements into user stories cards. The developers estimate how long it will take for the next release based on the business value and developing time. The cards are divided into individual tasks which will be assigned to specific programmers.
Pair Programming	Two programmers work together on one computer at the same time. One of the programmers is responsible to write the code while another one reviews that code. They switch roles periodically.
Test Driven Development	A programmer should write a test first. Then the test is run to ensure it requires new code to pass it. Next, he writes some new code then run the test again. The new written code should pass the test. The next step is to check if it can pass all the tests. After that, programmer should refactor the code, remove duplicates and clean the code. The whole process is repeated. It reduces bugs and ensures the correctness of the code. It can discover errors early with frequent delivery.

Table 3. Major techniques used in Extreme Programming (Measey, 2015, pp. 128-131; Beck, 2002, pp. 15-22)

Dynamic Systems Development Method (DSDM)

DSDM is another framework from agile processes that focuses on incremental delivery and strategic goals of business benefits by controlling time, quality, costs and risks. (DSDM Consortium, 2012)

Techniques	Description

MSCW	It is used for prioritising requirements. Below defines what the abbreviation stands for. Must have (M): user stories that must be delivered. Should have (S): if it is not delivered, it will cause significant problems to customers. Could have (C): if it is not delivered, it will just cause some problems. Won't have this time (W): The requirement may be added later or completely deleted with consent from customers.
Time boxing	It contains four stages. It begins with Kick-off to help understand time box objectives. It follows by investigation. Developers find out what they need to do to meet the objective. Then it comes to refinement which dominates 60 - 80% of the time box where the team should develop the products. After that, they need to double check everything is accomplished in the consolidation session. Finally, in the close-out session, the product is finished and accepted by the technical coordinator.

Table 4. Major techniques used in Dynamic Systems Development Method (Measey & Radtec, 2015, pp. 140-147)

Lean Software Development(LSD)

According to Mary Poppendieck (2003, pp. 13-14), there are seven principles in LSD: Eliminate Waste, Amplify Learning, Defer decision, Deliver fast, Empower the team, Build Integrity In, See The Whole. These principles aim to optimize the efficiency of a development team and minimize the wastes produced during the development process. The table below shows the techniques used in LSD.

Techniques	Description
Seeing Waste	The things that can't directly add value for customers is defined as waste. For example, partially done work, extra process and task switching. The development team should be able to identify waste.
Set-Based Development	It can provide better communication by communicating with constraints instead of choices. It requires far less data in order to convey more information. For instance, when the team wants to set up a meeting, the members should provide possible time constraints that suits everyone instead of choosing their own time slot and asking each person whether they are available.
Refactoring	Refactoring is a way to improve the code. It is expected that there will be some flaw in the first version. Through refactoring, the weaknesses can be identified and removed.
Pull Systems	Instead of pushing work to the members, pull systems let the team pull the work when there is capacity. It reduces cost for the company as it prevents overproduction.

Table 5. Major techniques used in Lean Software Development (Poppendieck, 2003, pp. 15-170)

Analysis

Major techniques in agile processes can be categorized into three main areas: Planning, Coding, Reviewing. They share more similarities than differences.

Planning

Planning Poker, Blitz Planning, and The Planning Game are the major techniques used in planning. They start at the beginning of an Agile process iteration. They have the same goal which is breaking down customer's requirements into individual tasks, then estimate the working time, order the priority and distribute the work to particular people. However, there are some differences between them on the process of achieving the goal. According to Cockburn,(2004, pp. 79-127) The Planning Game in XP uses user stories while Blitz Planning in Crystal uses tasks. Secondly, there is an assumption of fixed length iterations in The Planning Game but not in Blitz Planning. Also, there is no need to analyze the dependencies between user stories in The Planning Game whilst developers need to do that in Blitz Planning. The shortcoming of the Planning Poker is that it requires more workload and time. The team need to prepare a set of poker cards and devote much more time to make a consensus due to the complicated process.

Coding

After the planning section, the team starts building their product. It includes several techniques like Pair Programming, Side-by-Side Programming and Test Driven Development (TDD). The first two techniques involves work between two people. They check each other's code to review the blind spots created during the development process. They can share their knowledge, make immediate feedback and think in a different angle while coding. The difference is that Pair Programming is working on the same work but Side-by-Side Programming is working in their own works. TDD writes test before coding. It generates unit test cases and find the bugs in the early stages. These skills all aim to make frequent delivery, minimize bugs and detect critical errors as early as possible in order to increase the quality of work.

Reviewing

Through Reflection Workshop, Daily Scrum and Sprint Retrospective and Review, development teams can discuss and reflect whether the whole process is working smoothly. They can also find out what slows them down, speeds them up and any improvements to make throughout the process. In addition, it allows them to complete tasks more effectively in the next iteration.

Tools to Support Pair/Team Working

To help a pair or a team to organize, collaborate or communicate, there are several technical tools available. In general, they can be classified into low-tech and high-tech types that differs in technological use (Layton, 2012).

Low-tech tools

Basically, writing or drafting ideas on physical objects would fall in this category. Tools such as paper and pen, whiteboard and marker, as well as post-it notes are great tools to note down ideas and they are easy to discard and store. They are also low on cost in effort, time, and financial aspects. However, they may be exposed to physical hazards such as spilling of water or heavy winds. Although they may be easy to set-up, such tools are efficient only when members of the team are in the same physical environment or space.

High-tech tools

On the technical side, the following tools are available digitally and can only be accessed from electronic devices, mostly having access to the internet. In terms of communicating, they are categorised into a few types such as video conferencing, instant messengers, web-based desktop sharing, and collaboration websites (Layton, 2012). Other types of media such as email and social media can be used as well. High-tech tools minimise the risks of losing data as they can be saved on the cloud and be accessed from any other device and from any other location. They allow collaboration under the condition of internet access and electricity. If there is outage or technical problems on the devices, the collaboration will be disrupted.

<u>Video Conferencing</u>
Through available software programs such as Google Hangouts and Skype, a team can communicate by video calls to verbally express and also use the Share Screen features to display what they are talking about.

<u>Instant Messengers</u>
Current technologies that are available on mobile devices have a wide range of applications to communicate. These includes apps such as Whatsapp, Facebook messenger, Line, WeChat, and many more. The more commonly used messengers are Slack and Discord. These apps allow messages to be instantly sent and replied for quick communication and feedback.

<u>Web-based desktop sharing</u>
Sharing desktops allows team members to view the issues and updates that can be addressed immediately. This can be done through software applications or physically being at the desktop. Some examples of such applications are ScreenStream, SharedView and BeamYourScreen (Warren, 2017).

<u>Collaboration websites</u>
To allow communication and organization of ideas and development, there are many online tools that can be used so that data can be saved on the cloud. For version controls of a project, there are software programs such as Concurrent Versions System (CVS), Apache Subversion (SVN), Git and Mercurial to host a repository (Snyder, 2015). There are also bug trackers such as Jira and Bugzilla to ensure the integrity of a software development project. Besides that, there are online collaborative environments to allow members to work together in real-time such as Google Docs, Codeshare and Collabedit. Furthermore, project management tools such as Asana, JIRA, and Trello keeps track of the team member's roles and progress on the project.

Practices to Support Pair/Team Working

In order to work well as a pair/team, below are some of the practices to achieve this objective. (Measey & Radtec, 2015, pp. 69-90)

Face-to-Face Communication
It can be conducted both in person and through the internet. Team members share information and gives feedback immediately to correct misconceptions and work efficiently. Ideas can also be delivered precisely as group members can see the facial expressions and body movements which promotes better understanding. Also, it builds trust and bonding among the team with face-to-face interaction especially when the team is working with people that are from other countries. It is the most effective and efficient way to work as a team or pair.

Daily Stand-ups Meeting
It is a 15 minutes long meeting for tracking the progress of the project. Each team member needs to answer three questions: “What did I do yesterday?”, “What will I do today?”, “Are there any obstacles stop me from completing the work?”. The meeting is held by standing but not sitting because it can prevent people from falling asleep or other distractions. So everyone can focus and pay attention for this short meeting. It is an important practice for the team to know what they have achieved, plan for later work and more importantly to ensure everyone is contributing.

Retrospectives
It is always held at the end of an iteration. The team first needs to find out what went wrong and identify the problems. Then they should analyze the problems and come up with solutions for improvements. They can also make use of the solutions from the previous retrospectives. Moreover, the team should also reflect how they can work with other teammates and discuss if there are any improvements that can be made to improve productivity.

Synchronize Documentation
Apart from coding, documentation is an essential part in team working. The main types of documentation include code documentation, design documentation and test documentation. It provides comments, guidance and specification of the project. It is a good practice to synchronize the documentations when a new function is implemented such that every team members know what is going on and what new function is added by just looking at the documentations. It also allows external team to develop features that is related to the product.

Information Radiator
Information radiator is a display posted in a place where everyone can see when they walk past. It is usually posted on a wall or whiteboard. It promotes transparency. Everyone in the team can know the progress of the project and which person is responsible for the specific task. It is also a key element in daily stand-up meetings. Team members can present their works shown on board clearly and help the team to focus on the tasks that needs their attention. However, there is a limitation for the physical board which is it only works when all the members are in the same location. For a team that works across different countries or offices, they can use virtual boards that allow them to update the status on the internet.

For Second Coursework

Selections of Process, Techniques and Tools

In the second coursework, we are going to develop a game. It is a group work for six people which involves a lot of coding and documentation. Since Scrum is a management oriented methodology which mainly focus on organizing the work but lack of technical practices. Therefore we combine Scrum and Extreme Programming (XP) by implementing XP practices into Scrum. So that the project involves both technical and management techniques in order to increase the quality of the work.

As the coursework requires us to do team submissions every week, the sprint techniques in Scrum is the best way to satisfy the requirement. The duration of the sprint is one week depending on the tasks in the sprint backlog. Before the sprint starts, we should communicate with the customers and implement user stories and product backlog. It provides better understanding on the specific requirements and individual tasks that need to be achieved. Within the sprint, we will use pair programming technique. We are going to work as a pair and switch position frequently. So that we can share our knowledge and review on other’s work. We will also use test-driven development for coding tasks. It can help us discover some critical errors earlier in order to come up with solution before it is too late. We also select daily scrum as one of our techniques to track our progress. It is important because the second coursework involves six people. It can make sure everyone is contributing and doing their job by answering what they did, what they plan to do next, and what problems they are facing. After each iteration, we will conduct a sprint review and retrospective to review the work that we have done and plan the next step. We will also reflect on the last sprint to find out what should be improved. At the end of each sprint, we will also integrate our works to ensure it works as expected.

In terms of tools for the second coursework, the most important elements are communication and project management. In an agile environment, communication is crucial for product development and team collaboration. For general communication such as arranging time for meetups, Facebook Messenger can be used as most of the teammates would be able to respond promptly compared to other apps. As a secondary communication tool, Slack is recommended as a team chat software among the instant messenger tools listed above due to its integrated video conferencing and screen sharing features. The text-based chat and video chat functions plays a role in providing feedback on the code, seeking help, and clarification on project specifications on different channels such that different type of issues are separated. Furthermore, Slack has side integrations to other tools such as Github and Trello which will be used in the second coursework. In Discord, these side integrations are not provided and has to be integrated by bots that can take time to be installed despite its lightweight and convenient nature. Also, the screensharing interface is still in its beta phase but Slack has it fully integrated.

To manage our project, Trello can be used as it has a Kanban system that is easy to use and can be accessed remotely. Compared to other project management tools, the Trello board system describes tasks for the team, sets a deadline, and sets roles to members more efficiently. It provides file attachment and unlimited members, boards, cards, and lists within its Free plan. For version control, Github will be used in the second coursework. Although Bitbucket has a GUI interface that is easier to use, Bitbucket the Free plan limits the number of users to 5. This poses a problem as each team as 6 members. As such, Github is preferred as University of Bath has their own host server on it and the tool itself is widely used in industries so it would give us the experience of using it before entering the workforce. This will save us time on choosing a repository host and we will be able to locally host the code on our system. Most importantly, the tools chosen are free and can save us cost to develop the program for the coursework.

Risk Assessment

Following a risk management process as suggested by Moran (2016), we have discussed and applied his framework to present the following table (Table 6) that shows the possible risks with its likelihood of happening and consequences to bear. We also present a possible measure for the risk factors that will affect the second coursework. The likelihood and level of impact is ranked by the following:

(a) Likelihood: Almost certain, likely, moderate, unlikely
(b) Impact: Critical, high, medium, low

Description	Probability	Impact	Response
Uncooperative teammate	Low	Low	Try to communicate with the teammate or work with other teammates
Tools used for collaborative working are having issues	Moderate	Medium	Have backup tools ready in case some tools gets maintenance at time of use
File manipulation or database operations did not work	Moderate	High	Have other teammates review or ask the lecturer or tutor for help
Code passes the unit testing but product is not working the way it should be	Likely	High	Ask another pair within the teammates to review the code
Teammate does not know Java	Moderate	Medium	Find out other strengths of the teammate and apply them in other areas
Insufficient requirements	Almost certain	Medium	Ask the lecturer or tutor for clarification
Computer suddenly crashes and the code was not committed into repository	Low	Critical	Try to recover or rewrite the code
Lack of graphic design skills in the team	High	Medium	Examine alternative options such as royalty free graphics
Behind timeline compared to planned schedule	Likely	High	Periodically make sure work is consistent
Teammate is too sick to work on the project	Low	High	Spread out the workload of that teammate between other teammates
Natural disasters that will befall on Bath	Low	Critical	Follow protocols for city recovery and deal with the project at a later time

Plagiarism from other teams	Low	Critical	Protect codes from other teams and keep discussions to a high level
Server maintenance in hosting server	Likely	Low	Hosting can be done locally instead
Disagreement between teammates	Likely	High	Try to resolve the issues as a team or ask the tutors for advice

Table 6. Risk assessment of various factors that may occur in the second coursework

Pair Working Process

Description and Documentation

We have met at the lab sessions as well as for an hour out of class each week to discuss on our progress and also work face-to-face on this report. We have split the criterion as required between each other and worked on them separately (Table 7). Although the work was done separately, we have reviewed each other's work and commented on them. We have also made changes based on these comments and are satisfied with the progress. For this report, we have made use of Google Docs to collaborate as it provides real-time collaboration and there is a chat function in the tool itself for us to communicate. Furthermore, figures and tables can be inserted easily and styling can be done to headings. There is also a revision history that allows us to look back the changes we have made and can easily be reverted if required.

Week	Tasks
1	Created document on Google Docs, get contacts, search for sources on agile processes
2	Kimberley - Worked on CR1; Cheuk Hei - Worked on CR2
3	Kimberley - Worked on CR3; Cheuk Hei - Worked on CR4
4	Cheuk Hei - Worked on CR5; Kimberley - Worked on CR6, 7, 8
5	Complete requirements, final review and checks, submission

Table 7. Timeline of pair working process

Critical Analysis of Experience

During this coursework, we have found that it would be difficult to work alone due to the amount of reading and work to be done on the topics. Through pair working, we were able to reduce the workload and share the responsibilities. We also felt that we were more motivated when working in pairs than alone due to the need of clarification and communication as a group. Furthermore, we were able to brainstorm more ideas productively and express them more clearly after further discussions. We were able to learn from and correct each other. After reviewing each other's work, we were able to come to an agreement and produce satisfactory work. In hindsight, we think that working in a team helps to improve our communication and leadership skills by taking turns to lead in the coursework. Although we were able to cooperate well, there were some shortcomings such as occasional miscommunication and misunderstanding what we were trying to express. It took us a long time to come to a conclusion for deciding what to go with and there were times when we were both distracted from the coursework itself. In addition, there were times when we were lost about what the criteria of the coursework meant but we were lucky to be able to clarify and seek help from tutors. Through this experience, we learnt deeper about the subject together and were able to clear disputes that arose from uncertainty. In conclusion, it was an enjoyable experience working together as we were able to finish our report on time and were able to overcome the obstacles in completing this coursework.

Bibliography

Agile Alliance. *Agile 101* [Online]. Available from: https://www.agilealliance.org/agile101/ [Accessed 5 October 2017].

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, S., Sutherland, J. and Thomas, J., 2001. *Manifesto for Agile Software Development* [Online]. Available from: http://agilemanifesto.org/ [Accessed 5 October 2017].

Beck, K., 2002. *Test-Driven Development: By Example*. Boston: Addison-Wesley.

Beck, K., 2004. *Extreme Programming Explained*, 2nd ed. Boston:Addison-Wesley.

Berteig, M., 2014. User Stories and Story Splitting [Online]. Available from: http://www.agileadvice.com/2014/03/06/referenceinformation/user-stories-and-story-splitting/ [Accessed 1/11/2017].

Cockburn, A., 2000. *Agile Software Development*.3B ed. Boston:Addison-Wesley Longman Publishing.

Cockburn, A., 2004. *Crystal Clear : A Human-Powered Methodology for Small Teams.* Boston: Addison-Wesley Professional.

Coffin, K., Lane, D., 2006. *A Practical Guide to Seven Agile Methodologies, Part 2 : Page 2*[Online]. Available from: http://www.devx.com/architect/Article/32836/0/page/2 [Accessed 29 October 2017].

Craddock, A., Richards K., Tudor, D., Roberts, B., Godwin, J., DSDM Consortium, 2012. *The DSDM Agile Project Framework for Scrum* [Online]. Available from: https://www.agilebusiness.org/resources/white-papers/the-dsdm-agile-project-framework-for-scrum [Accessed 31/10/2017].

Iacovelli, A. and Souveyet, C., 2008. Framework for Agile Methods Classification. *Proceedings of the International Workshop on Model Driven Information Systems Engineering: Enterprise, User and System Models, MoDISE-EUS 2008 - Held in Conjunction with the CAiSE 2008 Conference*, 16-17 June 2008 Montpellier. CEUR Workshop Proceedings, vol. 341, pp. 91-102.

Layton, M. C., 2012. *Agile project management for dummies*. Hoboken: John Wiley & Sons, Inc.

Measey, P. and Radtec, 2015. *Agile Foundations - Principles, Practices and Frameworks*. Swindon: BCS.

Moran, A. Risk management in Agile Projects. *ISACA*, 2016, vol. 2, pp. 1-4.

Poppendieck, M., 2003. *Lean Software Development: An Agile Toolkit*. Boston: Addison Wesley.

Schuh, G., Wetterney, T., Lau, F. and Schröder, S., 2016. Next Generation Hardware Development: Framework for a Tailorable Development Method. *Management of Engineering and Technology (PICMET), 2016 Portland International Conference on*, 4-8 September 2016 Honolulu. Honolulu: IEEE, pp. 2563-2572.

Schwaber, K & Sutherland, J., 2013. The Scrum Guide[Online]. Available from: *https://www.scrum.org/resources/scrum-guide* [Accessed 31/10/2017].

Snyder, R., 2015. *Version Control Systems: Keep Your Code In Order!* [Online]. Available from: https://webinerds.com/version-control-systems-keep-your-code-in-order/ [Accessed 12 October 2017].

Snyxius, 2016. *How to implement an agile development process in easy steps*. Available from: https://www.snyxius.com/blog/implement-agile-development-process-easy-steps/ [Acccessed 5 October 2017].

Warren, G., 2017. *Top Computer Screen Sharing Applications* [Online]. Available from: https://www.lifewire.com/top-screen-sharing-applications-2377254 [Accessed 12 October 2017].

Williams, L., 2012. *What agile teams think of agile principles*. Communications of the ACM, 55(4), pp. 71-76.