**Project practice guide**

*The project practice guide identifies and describes the recommended practices for each of the following areas:*

**Collaboration & Communication**

**Requirements Engineering**

**Project Planning and Tracking**

**Managing Change**

**Quality Assurance**

*Links are provided to resources that support the practice and that enable future teams to upskill*

**Collaboration & Communication**

Agile methodology emphasises the importance of adaptability due to the dynamic and competitive nature of the modern software development, and in such an environment it is crucial that teams and team members keep communicating throughout the duration of the project to successfully collaborate the divided workload.

For our case study there is a team of 3 members, with a potential to grow into 18, and it would benefit them to follow certain practices that have been proven to improve communication and collaboration.

According to research from Agile modeling (n.d.) the most effective communication is face- to- face that are enhanced by a medium like a white board.

While communicating they should strive to utilise communication techniques that are applicable to their situation (e.g. if members are missing face to face conversation with notes may be more efficient), change your approach throughout the project to fit the situation.

Controlling the factors that may affect communication may also be a good practice.

**Physical proximity**

When people are closer together they have a greater chance for communicating

**Temporal proximity**

Whether people are working at the same time can also interrupt the workflow

**Amicability**

People’s willingness to hear and accept that thoughts given by another person

A widely used practice for collaboration include “daily scrums” where each team member report their progress and plan what they will be working on until their scrum the next day. It improves information sharing as well as awareness of what other team members are working on, how their work may clash with yours, etc.

http://innovation-regulation2.telecom-paristech.fr/wp-content/uploads/2007/05/DEDM13\_Can-Agile-Collaboration-Practices-Enhance-Knowledge-Creation-Between-Cross-Functional-Teams.pdf

Can Agile Collaboration Practices Enhance Knowledge Creation Between Cross-Functional Teams? (2012)

* SCRUM planning meetings
* Daily stand-up meetings
* Sprint review meetings
* Sprint retrospective
* Client demonstration meeting

http://agilemodeling.com/essays/communication.htm

**Requirements Engineering Practices**

The requirements engineering process is comprised of eliciting requirements from stakeholders, modelling and validating requirements (Hofmann, H., 2001, pg. 59). According to Hoffman, eliciting requirements is the most important and difficult process of a software project.

The following practices are recommended for successful requirements engineering from stakeholders by the Franchise group undergoing development of the Share portfolio management software.

**Verifying and validating requirements (not a practice but process)**

This process involves using modelling methods such as prototyping using gathered requirements, peer reviewing and doing walkthroughs of scenarios to improve the specifications of the software (Hofmann, H., 2001, pg. 65).

**Involve customers and users throughout the Requirements Engineering process**

By involving customers and users throughout the requirements engineering process, a software project can be more successful. Customer and user involvement allows a "better understanding of the real needs" (Hofmann, H., 2001, pg. 65) of a software project. Hoffman also states that the most important factor towards successful requirements engineering is user participation.

**Identify and consult all stakeholders and potential sources of requirements**

Identifying all stakeholders makes sure that all requirements are covered. By continually collaborating with stakeholders you can make sure that all requirements are collected and interpreted properly (Hofmann, H., 2001, pg. 65). This also allows projects to verify and validate requirements that may have been misinterpreted from individual stakeholders (Hofmann, H., 2001, pg. 63).

**Assign skilled team members and project managers**

Successful software projects have teams that are experts in knowledge of the respective application domain, IT, and requirements engineering ( Hofmann, H., 2001, pg. 65). Having the right ratio of these on a requirements engineering team will ensure that the best mix of knowledge, resources, and processes are used when engineering requirements (Hofmann, H., 2001, pg. 65).

**Allocate the most resources towards the requirements engineering process**

According to Hubert F. Hofmann in his paper, "Requirements Engineering as a Success Factor in Software Projects", projects that are more successful tent to have significantly more time and resources dedicated to requirements engineering than those that are not as successful. Hofmann states that between 15 to 30 percent of the total project effort should be spent on requirements engineering. The overall goal of this is that it allows a software project to develop a high quality specification of the software throughout its lifecycle (Hofmann, H., 2001, pg. 65, table 2).

**Maintain a good relationship with all stakeholders**

By maintaining strong healthy relationships with stakeholders allows a software project to be more successful. By continuously collaborating with stakeholders you can ensure that requirements are interpreted properly (Hofmann, H., 2001, pg. 65). Continuous communication also allows a software project to deal with changing requirements and avoid communication breakdowns (Hofmann, H., 2001, pg. 65). The biggest key benefit of maintaining a good relationship with all stakeholders is that it better satisfies the needs of the customers (Hofmann, H., 2001, pg. 65, table 2).

**Prioritize requirements**

Prioritising requirements is important to successful requirements engineering because of project time and budget constraints (Young, R., 2002, pg. 9). In many projects there is not enough time to provide the stakeholders with everything they want. By prioritizing requirements, unnecessary features can be minimized which reduces costs and design constraints (Young, R., 2002, pg. 9).

**Develop models and prototypes using the requirements**

According to Dr. Ralph Young in his paper, "Recomended Requirements Gathering Practices", prototyping in a software project is technique of building rough and quickly implemented versions of a system. Developing prototypes are important to making a successful product as they show the capabilities of the system to users and designers (Young, R., 2002, pg. 11). It allows serves to help facilitate communication by team members to help them understand the requirements better (Youg, R., 2002, pg. 11). According to Dr. Ralph Young also, "A Model is a representation of reality that is intended to facilitate understanding". Models are used in effective requirements engineering in order to help facilitate understanding of the requirements of a project.

The use of prototypes and models as a best practice help eliminate ambiguities and inconsistencies in a project's requirements (Hofmann, H., 2001, pg. 65, table 2).

**Use a traceability matrix**

A traceability matrix is a document that shows relations between two sets of values. In requirements engineering, a traceability matrix is used to track requirements' lifecycles from their conception through to their implementation (Hofmann, H., 2001, pg. 66). This is a best practice for requirements engineering, because it allows projects to document links between requirements and the products produced (Hofmann, H., 2001, pg. 65).

**Use peer reviews and team discussion**

Peer reviews and team discussion in requirements engineering is the process of conferring with team members and product managers to verify and validate requirements. This helps improve the specifications of a project and create higher customer satisfaction (Hofmann, H., 2001, pg. 65, table 2).

**Planning and Tracking Practices**

The use of SCRUM Sprint Planning meetings and artifacts would be key practices of SCRUM that can be used in planning and tracking. Artifacts include:

* Burndown Charts
* Sprint Burndown Chart
* Product Backlog
* Sprint Backlog

Because of the nature of this case study, having the first-half of the development process dedicated to redeveloping the current software, most requirements of the current software will not change. This would require a much more detailed process of Planning ahead and tracking that any potential functionalities are consistent with the requirements, otherwise this would force a chain reaction of problems in the future. The second-half of the case study is dedicated to extending the current functionalities, so a normal approach of planning and tracking in SCRUM would be required.

**Sprint Planning Meetings**

At the beginning of each Sprint a meeting, where stakeholders collaborate and plan what user stories to include in a sprint, so that all members involved know what is happening. Plans should all be resolved by the end of the meeting, so as to prevent as much hand-wringing as possible from the project. (Schwaber, 2004)

**Burndown Chart**

Burndown charts is a line chart that tracks the development’s work done to date. It shows the expected rate of work which was planned and puts it against an actual rate of work that has been completed so far. This will help for tracking the current state of the project, and if the team should increase their rate of work or not. The resulting chart shows the collision of reality with what was planned at the start of the project. (Schwaber, 2004)

**Sprint Burndown Chart**

A Sprint burndown chart is a sub-artifact of a burndown chart, where it will only show the rate of work that has been completed for only a sprint, by using an Effort value for user stories within a sprint. This resulting chart is a more highly-detailed burndown chart, that shows the amount of effort that is being done. This will help in tracking the work during sprints much more effectively, because for example user stories that are not having any work done will be displayed, which will cause a necessary action to be completed to make sure that work is done.

**Product Backlog**

A Product backlog is produced at the start of planning a project, which contains a collection of prioritized requirements which are done by negotiations between stakeholders. Although requirements will dictate how the project will be completed, the product backlog is dynamic, so change will be frequent. In other words, this product backlog will help plan future tasks to be completed with minimum changes to the plan, if no changes are of great size. (Schwaber, 2004)

**Sprint Backlog**

A Sprint backlog is produced at the start of each sprint, and is a sub-artifact of a product backlog, which contains only the user stories to be completed in a sprint, which are then each converted into tasks for the development team to complete. This will help spread out and plan the work needed to be done in a sprint.

**Managing Change Practices**

Choosing which practices to use for managing change are heavily dependent on the impact the change may have on the current development process. For changes with small impacts on the development, Daily meetings can be used to bring the issue and change to the team. For changes with large enough impacts that would stunt the development process, killing the current sprint and re-forming the sprint in Sprint Planning meetings would be most effective. However, for change that are large, but are not totally disruptive to the development process, adding the change to Backlogs can be effective for future development and planning.

**Daily Meetings**

For any small potential changes that may not have a huge impact on the development process that may occur, they can be easily managed with having daily meetings throughout the development process. Daily meetings traditionally consist of sharing with members of the team and stakeholders on discussing what was completed, any problems and what needs to be completed. This gives opportunities for any changes that have been encountered by anyone involved in the meetings to bring it up during meetings and have it dealt with accordingly, with everyone’s knowledge and approval.

**Sprint Planning Meetings**

For large impact changes that may cause the development process to be stunted, that may occur, can have the current sprint killed for it to be reset with Sprint Planning Meetings to start anew. Sprint Planning Meetings are where stakeholders collaborate and plan what user stories to include in a sprint, so that all members involved know what is happening. As an identified change has been detected to have stunted all development and discussed during a meeting, killing the sprint will be the most effective. Although this may cause delays in the plan, restarting a sprint with Sprint planning meetings will help re-plan the entire sprint relative to the change involved, so that development can be continued.

**Product Backlog**

For small changes and large changes that do not affect the development process , a product backlog can be used to track all changes, so that all stakeholders are able to view it and in the next sprint it can be brought up for discussion. A Product backlog is produced at the start of planning a project, which contains a collection of prioritized requirements which are done by negotiations between stakeholders. As “Scrum’s ability to respond to change has been one of its strongest assets in this environment and the backlog’s ability to evolve will make it a valuable addition to the team’s process” (Dulock & Long, 2015). So all changes would have no real impact on current development. All changes would be recorded for them to be easily tracked and for them to be brought up in the next sprint planning meetings to address them. (Hazrati, 2004)

**Quality Assurance Practices**

**Create a comprehensive project plan**

Creating a comprehensive project plan before starting development ensures that all requirements and planning stages are sufficiently documented and defined. The benefits of this are that mistakes and software defects can be minimised before development commences (Kevitt, M., 2010).

**Create a QA plan**

A quality assurance plan is a set of documents that outlines processes that ensure the quality of the product. It is a best practice to have a QA plan as the quality of the software can be defined and used as a basis for improvement in the future (Kevitt, M., 2010).

**Be comprehensive with test planning.**

A test plan is a document that describes how testing will be conducted and approached with regards to the resources, scope and objectives of a project (Hower, R., 2016). A test plan is very important to assuring quality of software as it outlines all team members and participants involved and the relationships between them and their responsibilities of testing (Kevitt, M., 2010). A test plan comprises of a number of documents including a test strategy guide, test plan and test cases. By defining all of these documents before testing, all resources can be efficiently allocated and all tools defined as well as picking up any potential issues before testing commences. This minimises costs and improves the quality of the software (Kevitt, M., 2010, pg. 21-25).

**Peer review all work done**

A peer review is a process where two or more team members collaborate on determining whether a deliverable meets the requirements of the project plan and is of a sufficient quality. Peer reviews are important to quality assurance as they help project teams evaluate each other's work and share knowledge which improves communication and minimises defects in software (Kevitt, M., 2010).

**References**

http://agilemodeling.com/essays/communication.htm

Hofmann, H. F., & Lehner, F. (2001). Requirements engineering as a success factor in software projects. IEEE software, 18(4), 58.

Young, R. R. (2002). Recommended requirements gathering practices. CrossTalk, 15(4), 9-12.

Schwaber, K. (2004). Agile Project Management with Scrum. Microsoft Press.

Hazrati, V. (2008). Managing Change requests in Scrum. Retrieved from <http://tolerosolutions.com/5-ways-agile-scrum-can-help-change-management/>

Dulock, M. & Long, H (2015). Digital collections are a sprint, Not a Marathon: Adapting Scrum Project Management Techniques to Library Digital Initiatives. Information technology and Libraries, Vol 34, No 4.

<http://dx.doi.org.ezproxy.aut.ac.nz/10.6017/ital.v34i4.5869>

Kevitt, M. (2010). Best software test & quality assurance practices in the project life-cycle. An approach to the creation of a process for improved test & quality assurance practices in the project life-cycle of an SME. Master of Science thesis, Dublin City University.

Hower, R. (January 2016). Software QA and Testing Frequently-Asked-Questions Part 2. Retrieved May 18, 2016, from<http://www.softwareqatest.com/qatfaq2.html>