Leassons Learned – EisMan project

Introduction

This document describes the retrospective results of the pilot project "Eisman App" within the openK consortium. The retrospective workshop was conducted November 10th 2015 with participation of clients and suppliers. The goal was to externalize lessons learned from this first project for future projects of the consortium. In general, some infrastructure topics, the development process, the requirements specification, and the estimation process were most discussed. This document focuses on lessons learned, as well as identifying open questions to be discussed, which might affect future projects. The document does not contain project-specific or vendor-specific topics.

The key findings of the workshop were:

- Even when working agile, an early common understanding of requirements is essential for the project success
- Minor and medium deviations in estimations can be compensated with "shared pain, shared gain", major must be early communicated and handled with change requests.
- It is essential to follow a proper Scrum process, including regular meetings and Sprints

In the following, these key findings as well as additional important lessons learned are described in detail.

Documentation and communication

The continuously productive and intense communication throughout the project duration was received very positively and as seen as a crucial success factor for future projects. The daily scrum meetings, in which even the product owner participated on a regular basis, were considered to be very useful. As a minor remark, the meetings could have been even more structured following the SCRUM guidelines, especially to communicate closed task in the daily meetings. The role of a Scrum master would have been beneficial to enforce this.

However, during the review it turned out that some misunderstandings have been occurring, which are related to both a not closed loop of communication and the way the documentation was provided. It must be ensured e.g. via feedback culture that any misunderstanding is cleared quickly.

It was reported that relevant documents were sometimes hard to find and access. They were created in different formats, mainly distributed via email, or stored in various systems. As such, the traceable documentation of changes and decisions was especially difficult. At the time, there were not any defined rules or guidelines, regarding which kind of documentation shall be created by whom, when, in which format, and how it shall be distributed. These rules and guidelines shall be defined for follow-up projects. This includes defined formats as well as guidelines on how documents are stored and versioned. The pragmatic and seamless

communications between the project partners was identified as a crucial criterion, as well as open formats, ideally modifiable with open source tools.

Summary:

- ⇨ Productive and continuous communication between all partners is essential
- ⇨ Daily SCRUM including the product owner were considered to be very helpful

TODOs:

- ⇨ Definition of communication and documentation guidelines (first version by PPC, later the QC takes over this task)

Eclipse Infrastructure / IP Process

The project and all its code artefacts are hosted on the Eclipse infrastructure and published under the EPL. All necessary agreements, especially on the suppliers' sides were signed immediately, which shows the strategic interest of the supplier. However, this seamless process must be ensured for future suppliers as well.

On project start and for the first weeks, the Eclipse infrastructure (Git and build server) was not yet ready to be used, the duration of necessary processes, such as the project proposal were not foreseen in the time schedule. Based on this experience, the critical path shall be analyzed. This information should enable future projects and committers to be prepared and necessary paperwork should be done early enough to have a running infrastructure at projects kick-off. As a fallback solution, projects can be hosted on any other open platform in the beginning (e.g. GitHub) and transferred, once the infrastructure is set-up.

The IP Process for some components took longer than expected (2-3 month). While a component is under review, it cannot really be used productively, as there is a risk, that it does not pass the IP review for license or IP related issue. In collaboration with the Eclipse Foundation, there should be a discussion on how this issue can be addressed for future projects. Besides potentially speeding up the process itself, a list of already reviewed components was considered to be helpful. The expected number of new frameworks to be introduced in the project is expected to be lower for future projects, as some common basic frameworks are already available. However, updating the version of existing frameworks was identified as another crucial case, as this might require several new IP reviews in parallel. Therefore, any potential version updates should be planed as early as possible. Currently, there is not any formal process defined for selecting additional technologies, yet. This shall be done by the AC, once it is installed.

There are open questions on how committers for existing openK projects get elected but also how they can potentially be removed from the project. This is related to quality assurance, as committers can potentially contribute unwanted changes. As of now, the plan is to follow the standard Eclipse processes, but in parallel, keep an eye on this issue, if any openK specific adaptations are required. As an example, sharing the project lead between supplier and contractor might be a reasonable way to share the control over a project. The quality committee might also come up with additional requirements for committers on projects based on their quality assurance plan.

Summary:

- ⇨ Necessary infrastructure, processes, and agreements should be communicated beforehand to suppliers and must be initiated in time to ensure a running infrastructure on project start

TODOs:

- ⇨ Capture the critical path for new committers and projects and provide guidelines on how early things shall be initialized in the future (BTC?)
- ⇨ Feedback session with the Eclipse Foundations on the described issues with the IP process, Speed-Up, "white-list", process for version updates (Steering committee?)
- ⇨ Process for selecting frameworks in the future (Architecture committee?)
- ⇨ Review the standard Eclipse committer process (Quality committee)
- ⇨ How to ensure the continuous quality in projects in an open ecosystem? (Quality committee)

Development Process

In general, the specification, communication, and common understanding of the requirements were identified as one the most crucial success factors for the project. The agile SCRUM process was received as positive. However, there were not regular Sprints during the regular project duration, but only two at a later time. Once regular Sprints were finally established, the results of those were meeting the requirements of the stakeholders. The general set-up, the exploratory character of the project and the domain posed some notable challenges to parts of the project's development process. The corresponding "lessons learned" are summarized in the following paragraphs, based on the affected activity.

Requirements Specification

The requirements specification was done in three phases, the first two are in preparation of project, the third is during the project is delivered. This process is also relevant for future projects:

1. First the group of sponsors specified an initial version of requirements.
2. The initial requirements were presented to and discussed with potential suppliers. Based on their queries and feedback, the requirements were refined to the final call for proposals. Based on this, suppliers provided their proposals. I has been pointed out during the retrospective workshop, that by submitting a proposal, supplier reflect and communicate their detailed understanding of the given requirements.
3. Finally, after a proposal has been accepted and the project started, the requirements were specified in detailed in an agile process and with close communication between the supplier and the product owner.

It was pointed out that is necessary to achieve a common understanding of the overall system as well as a common vocabulary. Therefore, even when working in an agile process, the backlog should be commonly understood. This also helps to identify complex requirements, which should be planned in early sprints to reduce the risk.

Early UI mock-ups were increasing a common understanding of requirements. Furthermore, the introduction of state or BPNM diagrams for specifying use cases significantly reduced misunderstandings. This technique should be used for future projects, as well. It was discussed to use BPMN for an initial requirements specification in the future.

Summary:

- ⇨ An early common understanding of the overall system is important, even when working agile
- ⇨ Specifications in BPNM were considered to be very useful
- ⇨ Early UI Mock-Ups help to get a common understanding of use cases
- ⇨ Complex requirements should be identified and realized as early as possible

Estimation

The general challenge posed by a „fixed price“ project combined with an agile approach was discussed. While there is a total estimation in the beginning of the project, requirements are detailed and potentially changed later on. The “shared pain, shared gain” approach compensated for small and medium deviations, but did not work for requirements, which significantly changed the initial overall estimation. For those, change request should be submitted as early as possible.

There was a discussion about how to come up with more precise overall estimations at the proposal phase. For some use cases, test data could provide more insights to estimate the complexity. More suggested approaches are: Additional workshops for detailing requirements, external review of estimations, and external estimation by third-parties before the call for proposals is done. On the client side, estimating the value of a system before doing a call for proposal was also considered. Finally, workshops or dedicated projects could help to identify complex requirements before the proposal phase, this would enable one to point out unrealistic estimations for certain requirements. Furthermore, it should be made explicit in proposals, if certain requirements are planned to be implemented as “platform” components, e.g. the CIM model, as this requires more effort and affects the estimation.

During the project, clients pointed out that an ongoing status update on the estimations of items in the backlog is important to track the general project scope, to communicate deviations and to identify the need for a change request. Therefore, there should be a regular review of the original estimations of backlog items, even if they are not yet planned in the sprint. Any deviation should be communicated as early as possible to get an early awareness of any risks

Summary:

- ⇨ Minor and medium deviations in estimations can be compensated with “shared pain, shared gain”, major must be handled with change requests
- ⇨ During the project, backlog items should be continuously reviewed about their original estimations, any deviations should be communicated

TODOs:

- ⇨ Discuss potential refinements in the estimation process during the proposal phase (Steering Committee?)

Sprint Review

The continuous delivery of a demo able product at the end of every sprint was identified as an essential success factor for any future project. It allows the product owner to compare the results with the given requirements as well as to provide early feedback and therefore reduce the risk for misunderstandings. However, the requirement for an on-going runnable product can collide with the implementation of cross-cutting requirements. It needs to be clarified if this is a one time issue due to the fact that this was the first oK module or if the same issue may occur in future, too, until a complete data model will be available. As an example, in the pilot project, when the data model (CIM) was initially specified, when there was not yet any UI or demo able feature. This data model was considered to be one of the most valuable outcomes of the overall project, as it can also be used as a basis for future projects.

However, it was discussed, if even these kinds of basic artefacts should be developed in a more iterative way based on vertical slices, even if this would require additional refactoring efforts later. This would help to ensure common understanding on the currently implemented requirements. The ability to have a runnable and demonstrable system at the end of a Sprint must be explicitly considered when planning Sprints as it influences the priorities of the product owner. Ideally, there is a precise description of the expected demo scenario. After a Sprint, the system must be explicitly reviewed by the product owner. This procedure requires a running project infrastructure including a running build from day one, as well as a set-up demonstration system for deploying the results of every Sprint. The ability to have demo able slices should even influence the scope of new projects. However, it was agreed, that even though continuous delivery shall be the default for all future projects, there might be some cross-cutting concerns, which will require deviations to this. One suggestion to deal with these problems was to extract these concerns into separate projects. In this case, intermediate results must still be carefully reviewed after every Sprint.

Summary:

- ⇨ Runnable slice shall be delivered after every Sprint by default and from the beginning
- ⇨ This must be explicitly considered by the product owner when planning the Sprint
- ⇨ Infrastructure for delivery and demonstration must be available from day one

Miscellaneous and open questions

- ⇨ Project resources should have overlapping tasks to reduce the risk of a drop out of team members.
- ⇨ Any test data required from clients should be identified and requested as early as possible, as providing the data usually takes time.
- ⇨ How and when will the system be deployed in a productive environment?
- ⇨ Where exactly is the border between the openK platform and SCADA systems?
- ⇨ How can the developed CIM profile be published, ideally under an open source license?