0.1 Analysis of the GIRAF project process

During informal discussions within the group, both during the first and second sprint review, we agreed the process used for the Graphical Interface Resources for Autistic Folk (GIRAF) project is probably the cause for many of the issues the project suffers from. This section offers an analysis of the software development process of the GIRAF project, which can be used for process improvements on this or future semesters.

The software development process of the project is an agile process that heavily leans towards Scrum. Issues pointed out in this section are mistakes in the context of Scrum or the practices used that are likely the cause of some of the issues brought up at the sprint review and project meetings.

0.1.1 Dividing by responsibility, not functionality

Large projects require a scaling of the Product Owner (PO) role which is done by introducing a hierarchy of collaborating POs, shown on Figure 1[2][5]. Mike Cohn uses the phrase "Sharing Responsibility, Dividing Functionality" to describe the structure of the hierarchy. Dividing by functionality means that all teams work on some functionality that is delivered to end users. Figure 1 shows an example of such a division of an office suite. One group works on the functionality off unctions and another on the graphs of the Spreadsheet product line [2][5]. Alternatively, the division could be made by responsibility. Some teams would have the responsibility of the design of the graphical user interface, whereas others have responsibility for the component managing storage and opening of files. Yet another team may be responsible for testing the different components.

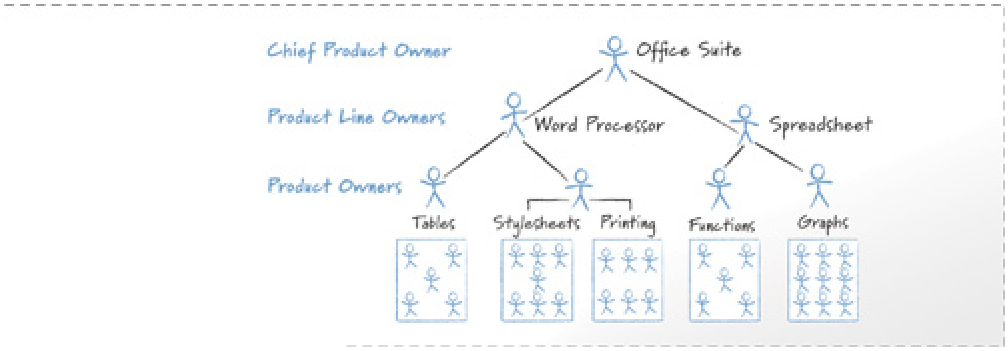

Figure 1: The project owner hierarchy on a large project[2].

In the GIRAF project division is made by responsibility. The Database (DB) subproject has responsibility for the OasisLib component for instance. Rather than viewing the project as a single product, the OasisLib component is instead seen as a separate product with the users primarily being the groups of the Graphical User Interface And Features (GUI) subproject.

The issue with this division of responsibility became apparent when formulating user stories.

The user stories handed to the DB subproject were tasks passed on to the DB subproject. Additionally, displaying value to our customers became almost completely trivial. Our customers, the teams working on other applications, would already have integrated our solution by the end of the sprint and thus had already seen the value of the solution.

This division has probably also existed in the previous semester, since it is unlikely that the developers of the GIRAF applications would have chosen the implementation discussed in Section **??**, because it makes querying the database quite cumbersome for the users of `OasisLib`. Additionally there has been a few complaints about using `OasisLib` for querying the database in reports from the previous semester, at the latest sprint planning and from group 12 with whom we are collaborating with during this sprint.

The obvious solution to the issue with dividing by responsibility is to make the division by functionality instead like the example shown on Figure 1.

0.1.2 No real Product Owner in the project

While it is generally a good idea for end users to be POs, because they are among the key stakeholders and as such have great knowledge of the problem domain. The PO role itself requires a high degree of availability, as well as attention to the interests of other stakeholders. The PO should also be a single person, to avoid conflicting messages, not a group ofi ndividuals[3][9].

The top level Product Owners (topPOs) show up for the sprint planning, prioritizing the product backlog by choosing which backlog items will be worked on in the current sprint. During the sprint itself, they are not available every day, although they can be contacted by mail. Additionally, there is another important stakeholder, namely the government, which has certain requirements about the protection of personal data in systems used in public institutions. This requirement represents a security concern that is often affected by the architecture of the system. The longer the requirement is ignored, the more cumbersome it will become to change the architecture. Furthermore, meeting this requirement is vital to the success of the GIRAF project, since it cannot be used in the public institutions of the topPOs, yet the topPOs have given it very low priority.

The responsibility of this potentially critical misprioritization does not rest on the topPOs alone, however, since the developers also have a responsibility to help the topPOs prioritize, though the topPO has the final word on the matter. Furthermore, the exact responsibilities of the topPO role have not have been presented and/or defined to the topPOs. Finally security concerns are non-functional requirements, which should not be captured by user stories and prioritized at all. Rather they should be listed as constraint cards or some other description of non-functional requirements[3]. It should be mentioned that security is just one of many alike constraints formulated as user stories.

Each subproject has their own sub-project Product Owner (subPO), however they let the users prioritize the product backlog and do not themselves understand the requirements sufficiently for developers(other groups) to get elaborated details about them. Although user stories are reminder of a conversation, the product owners at different levels are either unavailable or does not understand the needs of theirs users in depth. In addition to this, user stories do not provide the necessary details to implement them. For these reasons, the use of user stories does not make much sense under the current project conditions[4][3]. This issue is also accentuated by the lack of acceptance criteria commonly associated with the use of user stories, and each group does not have a uniquely identifiable and dedicated PO[3][12] [2].

0.1.2.1 Possible solutions

A solution to this issue could be to make the topPOs aware that they are required to be much more available and to make subPO aware that they need to be able to answer all questions about the users needs. The topPOs are external customers and it may not be possible for them to be as available as they are required to fill the topPO role. The issue with a customer or user being unavailable for a majority of the time is however not new and a common solution is to use a user proxy, which is a person that is not a real customer or user, but functions as a PO or on-site customer for the project. The issue with this approach is that developers make poor user proxies[3].

Another issue regarding the use of user proxies that is more specific to the conditions of the GIRAF project is that except for the users and the semester coordinator, the entirety of the staff working on the project is replaced after the duration of a semester. This means that each year a new group or person needs to become a user proxy and spend a lot of time understanding the users' needs. A suggestion to solve this might be to spend some time documenting the needs of the users, however this defeats the purpose of writing user stories in the first place - to avoid detailed documentation of requirements[3].

User stories are not however the only way to document functional requirements in an iterative and incremental development approach. In Unified Process, a use case model is used[7] and in other agile approaches, including Scrum, use cases can be used as well. The use cases in this situation has the advantage that they are much more detailed and are intended to be part of more elaborate documentation. Just like user stories they evolve over time, however they cover more functional requirements than user stories, which may pose challenges when dividing work. Additionally, user stories are easily understood by users or POs whereas use cases are more technical and possibly harder to understand[4][11][3].

0.1.3 The product is seeing very limited use

One of the tenets of agile development is to deliver working software at the end of every sprint, which means a potentially shippable product increment[3][6].

The project has been in development since 2011, yet it is not used regularly in the institutions according to key stakeholder Mette Als, one of the users. While only a single semester is spent on development each year, the core features described at the beginning of the project by Mette appear to be quite simple, but none of them have been entirely finished. These simple features were the ability to find and display pictograms, displaying sequences of pictograms to illustrate steps of an activity and a week schedule to reduce the overhead of changing their current physical week schedules.

The apps `WeekSchedule` and `Sequence` which are the apps implementing these features are however still being developed during the current sprint. Several lesser essential apps, such as the game apps, have been developed even though the more critical apps still lack essential features such as the ability to save the schedule.

This issue is in part due to the lack of PO involvement described in subsection 0.1.2. With only user stories and no PO to go to for a more detailed description of the story, some requirements from the PO are bound to be missed or overlooked. Additionally, the students developing the software project are not used to work together and show a preference to work on their own individual modules. This preference has likely lead to some of the features, which were never requested by the customer, such as the use of QR-codes, and to the incompleteness of the core functionality(By grabbing lesser important backlog items to work on).

Additionally the security concerns regarding the Danish data protection legislation have yet to be resolved. This means that the GIRAF project cannot be integrated into the institution.

While the non-functional requirements should have been built into the system from the beginning, the next best solution is to ensure it as soon as possible. Additionally, defining a minimum viable product (MVP) could be of value for the project. A MVP is the product that has exactly those features that allows it to be deployed. This is not just a potentially releasable product, but a product that is actually intended to be released to at least a subset of possible customers or users[13][1]. Once a MVP is defined, it should be implemented and deployed as fast as possible to the environment in which it will be used. An example of a MVP for a web shop could be a website where the user can browse items by title and buy each item individually.

0.1.4 Only features and bugs on the product backlog

User stories or use cases describing features or bugs are likely to be the most numerous type of backlog items appearing on a Scrum product backlog. There are however other items, such as technical work and knowledge acquisition items as well. Technical work items could be configuring Jenkins for continuous integration, setting up a test environment or major refactoring of some part of the system. Knowledge acquisition could be researching different libraries or technical solutions. These items may not directly provide business value, but can be necessary tasks. The PO and developer team should collaborate on prioritizing these items. The developer team should explain the value to the PO, and the PO must choose when it is convenient to perform these tasks as with any other backlog item[10][8].

An example of this could for instance be by swapping out the database system to enhance portability. The PO could choose to postpone the technical work in favor of getting more total amount of user stories completed during the upcoming sprint.

In GIRAF the product backlog consists only of user stories and related tasks. The Build And Deployment (B&D) subproject is assigned technical work exclusively, such as setting up Jenkins. Rather than treat these stories as technical work to be prioritized by the PO, the B&D subproject is treated as a project making a product of their own with complimentary user stories. This view is identical to that of the DB subproject as discussed in subsection 0.1.1 and causes an over-emphasis on the type of technical work handled by the B&D sub-project. The GUI and DB sub-projects implement features and tasks, deemphasizing or ignoring potentially very valuable technical work or knowledge acquisition. In any case, the decision on whether or not to complete features or technical work is taken away from the PO.

Combining the division off unctionality described in subsection 0.1.1 with the addition of technical work and knowledge acquisition items on the product backlog will give back the decision making power to the PO.

0.1.5 Summary

The project is currently divided by responsibility, rather than functionality. This leads to artificial user stories from the different subproject groups to each other as if they were building separate products. The project should ideally be redivided by functionality not responsibility.

Another issue the project suffers from is the lack of a proper POs. The POs of the GIRAF project as a whole are not sufficiently available to fill the PO role. The subproject POs are available, but do not possess the necessary domain knowledge to explain or prioritize user stories and thus cannot fill the PO role. The user stories additionally lack acceptance criteria accentuating the issue. The user stories do not themselves describe the necessary detail to implement the feature they describe and having POs available to supply this detail is vital.

Solutions to this issue includes establishing a user proxy, however the obtained knowledge would be lost from semester to semester unless documented. Another solution is to stop using user stories and switch to use cases, which can be used for agile development as well. Use cases are generally larger than user stories and it may therefore be harder to divide work and may not be as easily understood by POs or users.

GIRAF is seeing very limited use. This is in part due to the lack of compliance to the Danish data protection legislation and part due to core features not being fully completed. The cause of this incompleteness likely stems from the lack of true POs and the students developing the system. The students are used to working in separate groups and not collaborate with other groups and thus show a preference to work on separate modules, perhaps artificially creating work. A definition of a MVP for the GIRAF project could give the project direction and provide valuable experience from the deployment of the system.

Only user stories and related tasks appear on the GIRAF backlog, which de-emphasizes the value of technical work and knowledge acquisition for the GUI and DB subprojects. Meanwhile the B&D subproject does nothing but technical work, perhaps over-emphasizing the importance of the type of technical work they are responsible for. Through all of this the PO is left out of the decision making.

Bibliography

[1] Vladimir Blagojevic. The Ultimate Guide to Minimum Viable Products. Website, 2013. `http://scalemybusiness.com/the-ultimate-guide-to-minimum-viable-products/`.

[2] M. Cohn. *Succeeding with Agile: Software Development Using Scrum*. Pearson Education, 2009.

[3] Mike Cohn. *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.

[4] Courtney. Use cases vs user stories in agile development. Website, January 2012. `http://www.boost.co.nz/blog/2012/01/use-cases-or-user-stories/`.

[5] Colin Bird & Rachel Davies. Scaling scrum. Presentation, 2007. `https://www.scrumalliance.org/resource_download/287`.

[6] Kent Beck et. al. Manifesto for agile software development. Website, February 2001. `http://agilemanifesto.org/`.

[7] Craig Larman. *Agile Iterative Development: A Manager's Guide*. Addison-Wesley, 2004.

[8] Ben Linders. Should You Create User Stories for Technical Debt? Website, March 2013. `http://www.infoq.com/news/2013/03/user-stories-technical-debt`.

[9] Scrum Methodology. Scrum Product Owner. Website. `http://scrummethodology.com/scrum-product-owner/`.

[10] Mountain Goat Software. Scrum Product Backlog. Website. `http://www.mountaingoatsoftware.com/agile/scrum/product-backlog`.

[11] Matt Terski. Agile Use Cases in Four Step. Website, September 2009. `http://blog.casecomplete.com/post/Agile-Use-Cases-in-Four-Steps`.

[12] Don Wells. Acceptance Tests. Website. `http://www.extremeprogramming.org/rules/functionaltests.html`.

[13] Wikipedia. Minimum viable product. `http://en.wikipedia.org/wiki/Minimum_viable_product`.