1 Software Development Method

One of the great challenges this semester is to coordinate the development process across all 14 groups. To ensure a smooth process we define the development method of the multi-project.

Chapter Organization The remainder of this chapter is organized in the following fashion:

- in Section 1.1 we describe how the multi-project is organized and introduce the hierarchy in the project;
- in Section 1.2 we detail the responsibilities delegated to most groups.

Chapter Abbreviations This chapter introduces the following abbreviations:

GUI	Graphical User Interface	**B&D**	Build & Deployment
DB	Database	**PO**	Product Owner

1.1 Multi-Project Organization Method

The multi-project consists of 14 groups. Two groups of two and the remaining groups consists of four people each. All groups collaborate on building the Giraf applications. The groups are organized into three subprojects: *Graphical User Interface* (GUI), *databases* (DB), and *build and deployment* (B&D) by recommendation of the semester coordinator.

The GUI groups deal with bugs and implementing new features in the front-end apps. The DB groups manage and maintain the database. The B&D groups control the tools supporting the building and testing environment as well as the deployment of the apps.

The multi-project groups work with an iterative development method, and the semester is divided into four iterations. No groups have any prior experience with the code base at the start of the project, nor with the requirements of the users. Therefore there are many uncertainties associated with the multi-project. This makes the application of an agile method suited for the project. Suppose the multi-project is organized in a traditional method. Then the multi-project groups will spend much time initially analyzing the current code base and future requirements, rather than actually work on making the code base work (a prime wish for the external customer). These requirements may change over time, which makes it difficult to deliver what the users want.

The agile method used for the project is Scrum, and the groups are organized as Scrum of Scrums. Scrum of Scrums divide the developers into several layers of Scrum teams

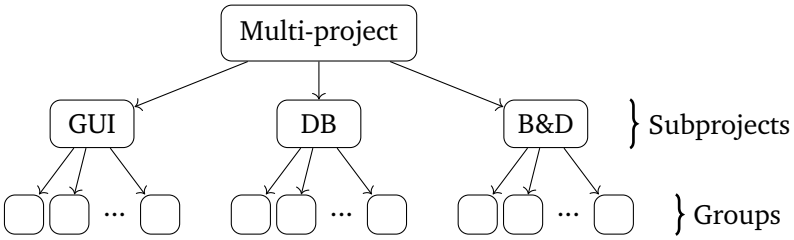

Figure 1.1: Illustration of multi-project organization

[8]. Figure 1.1 shows the structure of the multi-project organization, which consists of three levels:

Multi-Project Level The purpose of the overall level is to ensure that the roles, work products, and meetings are followed as intended. That is, that the development method is followed. A weekly meeting is held where the status of each subproject is discussed, and the development method and roles are evaluated. Every group has at least one representative at the weekly multi-project meetings.

Subproject Level Each subproject has the responsibility of solving a number of user stories related to either GUI, DB, or B&D. This level follows Scrum of Scrums. Representatives from each group in a subproject have sprint planning and sprint end meetings with the other members of the subproject and hold at minimum two Scrum meetings a week. We chose not to meet every day because we think that it will steal to much time, compared to the benefits. It is common for such a meeting to be held two or three times a week [8].

Group Level Each group solve some of the user stories of their subprojects. While Scrum of Scrums dictates that Scrum is followed at all levels, this is not the case for this multi-project organization. This is of course a major deviation from Scrum of Scrums. When initially deciding the software development method, some groups expressed an interest in not running Scrum. Thus, each group is given the freedom to organize how they see fit. However, all groups are expected to fill in the work products and attend the meetings described later.

1.1.1 Work Products

Two work products from Scrum are used, and every group is expected to help updating and maintaining these:

Product Backlog The product backlog contains the items of value to the customers, typically features, but also other items [4]. On the multi-project level, there is a product backlog containing all items for the entire multi-project. The purpose of the multi-project level product backlog is to maintain the product backlog for the

entire multi-project. The items in the product backlog are categorized according to each subproject. This enables each subproject to manage their items themselves, removing unnecessary management from the multi-project level.

Release Backlog The release backlog contains the items that must be finished by the end of the current sprint [4]. These items are selected from the product backlog during sprint planning meetings that each subproject hold. Each sub-project has their own release backlog, which is a subset of the multi-project release backlog.

Since it is not required for each group to use Scrum at the group level, there is no common sprint backlog, i.e. a collection of tasks shared among all groups.

The product backlog and release backlog contains the following categories of items:

Feature Functional requirements (features) of the product are represented as user stories, as they are lightweight and fit well into the agile principles [6].

Bug Because bugs also represents behavior the user wants to be changed, they are considered no different from features [7], and so are represented as user stories.

Constraint A way of handling non-functional requirements, while not forcing it into a user story, is to turn the non-functional requirements into constraints [3, ch.16]. Most often, these constraints are related to e.g. performance, usability, and security. Constraints can either be written on separate constraint cards [3, ch.16], or written in the corner of a user story card if it is relevant only to that single user story [3, ch.7]. Due to the structure of multi-project, it will make sense for us to introduce two-level constraints. Multi-project constraints are relevant for the entire project as a whole, for example *the software is written in Java* or *all apps must be easy to internationalize*. Some constraints only apply to some areas of the project, and these are handled by the individual groups. Examples of these are *the initial database sync must not take more than 5 minutes* or *the user manager must only sync data of the user logged in*. These constraints are written on either a constraint card or on the relevant user story.

Knowledge Acquisition Since no one in the multi-project has worked on it before, and because the project is complex, sometimes it is needed to investigate things to know it if is worth investing time in. They can be formulated as user stories, but do not need to [6]. Knowledge acquisition items are handled the same way as user stories: They are prioritized, estimated, etc. Estimation can be hard when investigating an unknown subject. The item can be investigated just enough to do further estimation by *timeboxing* an initial investigation. This practice is in XP called a *spike* [3].

Technical Work Sometimes effort has to be spent on something the users are not directly interested in, e.g. updating software on the developer machines, updating the database software on a server, or refactoring parts of the code. These items can be part of the product backlog [3]. They can also be formulated the same way as user stories [6] and are handled the same way.

The backlog items *features* and *bugs* are represented as user stories, whereas *knowledge acquisition* and *technical work* can be represented as user stories, but do not need to.

Cohn [2] presents the following template for a user story:

As a ⟨*type of user*⟩, I want ⟨*some goal*⟩ so that ⟨*some reason*⟩.

This template is used for user stories in the multi-project. If the user story is suggested by developer, the source of the user story should be noted in case there are questions about it.

A user story must be sufficiently small for it to be completed in one sprint [2]. A user story must also have *conditions of satisfaction*. These are high-level acceptance tests for the user story and must be true for the user story to be completed. A user story can have multiple conditions of satisfaction [2]. User stories are prioritized by the product owner (possibly with help from the customers) and select the user stories for the next sprint [4].

1.1.2 Roles

Two roles from Scrum are used in the multi-project:

Scrum Master On the multi-project level, we are responsible for the development method. This means that we act as Scrum masters on the multi-project level. On the subproject level, one person is charge of the meetings — they make sure that the Scrum meetings on this level are held. On the group level, each group is free to do what they see fit.

Product Owner The product owner (PO) is responsible for maintaining contact with the customers and being their representatives. It can be difficult to have a single PO with an external customer as customer in the multi-project. Therefore, the B&D and DB groups often do not have the external customer as their direct customer. Rather, these groups help the GUI groups achieve their goal by building solutions for them. The external customer is often not interested in the implementation details of build and database interfaces.

As such, we decided during sprint 1 that there are three POs in the multi-project, one for each subproject. The customer for the GUI PO is the external customer, whereas the customers for the DB and B&D POs are the groups of the other subprojects, as illustrated in Figure 1.2.

1.1.3 Meetings

Each week there is a meeting at the multi-project level. At this meeting the status of all groups in the multi-project is presented and the development method is evaluated. This ensures that all groups know the current status of the project. This meeting differs from a regular Scrum meeting in that it includes a point on method evaluation, so that the development method can be continually improved.

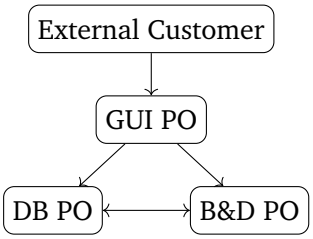

Figure 1.2: Illustration of product owners

Each sub-project holds internal Scrum meetings as well. These sub-project Scrum meetings are held at least twice a week. The sub-projects organizes these meetings themselves and use them to discuss things that are relevant only for the members of the sub-project.

Each group decides how to manage their internal affairs. As long as they produce the required artifacts, they can decide for themselves how to organize.

Since there are three product owners in the multi-project, three sprint end meetings and three sprint planning meetings are held between sprints. These meetings were proposed by us, following the recommendation from Bird and Davies [1], and afterward approved by the other groups.

Sprint Planning Meetings The purpose of the sprint planning meetings is to ensure that the product backlog is up to date and to select user stories for the release backlog.

GUI At this sprint planning only one representative from each GUI group participate as *pigs* (people who are allowed to talk), as well as the GUI PO and one representative from the other PO groups (although they have a less important role at this meeting). All other people in the multi-project are *chickens* (people who are not allowed to talk). This meeting is held with the external customers. By making it so that primarily the GUI groups are pigs, the external customers are abstracted from internal development such as continuous integration and database management.

DB and B&D There are separate sprint planning meetings for DB and B&D, where each subproject establishes the needs of the other subprojects. These meetings are internal, and the external customers do not participate, since DB and B&D do not have these as direct customers. DB and B&D will primarily have the GUI groups as their customers, who need various services from the DB and B&D subprojects, in order for them to fulfill their own user stories directly related to the external customers. However, it might be that DB needs something from B&D and vice versa. Again, this abstracts the external customers from internal development.

Sprint End Meetings At the sprint end meetings, the work done in the sprint is presented to the customers.

GUI At this meeting the external customers are presented with the work of the GUI groups. Only the GUI groups participate as pigs at this meeting.

DB and B&D DB and B&D each hold a separate sprint end meeting where they show the groups in the other sub-projects what they have done in a sprint. The external customers are not present at these meetings, so that they are not bored with internal details. The GUI sprint planning meeting is separated from the DB and B&D sprint planning meeting by a few days, such that the product owners have time to organize, and make the requirements propagate from GUI to the other sub-projects.

1.1.4 Office Hours

A vital part of working agile involves being able to easily communicate with all members of the multi-project. To ensure this all groups must be reachable on e-mail all weekdays 09:00–15:00, and preferably be in the group rooms. An exception for this is when there are courses.

1.1.5 Tools

To support the development method of the multi-project, an online project management tool, Redmine [5], is used. This tool was in use the previous year, and so this year continues to use it. The main features are:

Wiki A wiki is used for various information, such as guides, how the development method is structured, meeting summaries, etc. Previous years had several separate wikis for each subproject. This year there is a single wiki providing central information.

Forum The forum is used for non-crucial communication. It is mostly used for non-work related discussions such as social events. Groups are encouraged to communicate to each other directly by going to each other's group rooms, rather than using the forum.

Backlog Redmine has a issue tracker tool which we have repurposed as a backlog. The tool is limited, and forces to have all the backlogs in the same place as well. The user stories are also mixed in with tasks, support issues, meetings, and the other items in the backlog. Filters can be used to isolate specific items, such as user stories in the product backlog.

1.2 Group Roles in Multi-Project

The multi-project contains responsibilities that have been assigned to the groups within the multi-project. These roles include management of Redmine, server, Git, Jenkins and other various tasks. Following they are shortly described:

Redmine — General Maintaining users, roles, and plugins on Redmine as well as managing work across the other Redmine groups below.

Redmine — Wiki Structuring and maintaining the wiki. They also read the content and make sure it is adequate, and notify the relevant groups if problems are found.

Redmine — Forum Structuring and moderating the forum.

Redmine — User Story and Task Tracking Structuring and maintaining the user story and task tracker as well as helping people creating the items in the tracker. They also maintain guidelines for task creation.

Server Maintaining the server software and controlling access to the server. They install and upgrade software upon request and help solving issues requiring root access.

Git Maintaining the Git repositories and controlling access to these. Furthermore, they issue guidelines for Git usage.

Jenkins Maintaining the Jenkins setup. This is the responsibility of our group, and as such will be detailed in this report.

Process Supervising, developing, and refining the process method. This is also the responsibility of our group, and will as such also be detailed in this report.

Code Style Developing and encouraging a code style across all code.

Customer Relations Maintains customer relations. They review and send any email sent to the external customers.

Web Administrator Maintains the public project website `http://giraf.cs.aau.dk`.

Android Guru Persons with experience developing Android apps, and who can be asked for help.

Google Analytics and Google Play Maintaining the Google Play listing of the software and generating guidelines for how to get crash reports from Google Analystics. Furthermore, they published the builds during the first sprints. .

Graphics Developing a graphical style and enforcing this as well as creating the graphics upon request.

Bibliography

[1] Colin Bird and Rachel Davies. *Scaling Scrum, Scrum Gathering London 2007, slides*. 2007.

[2] Mike Cohn. *Succeeding with Agile*. Addison-Wesley Professional, 2009. ISBN: 0-321-66053-6.

[3] Mike Cohn. *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional, 2004. ISBN: 0-321-20568-5.

[4] Craig Larman. *Agile Iterative Development: A Manager's Guide*. Addison-Wesley Professional, 2003. Chap. 7. ISBN: 0-13-111155-8.

[5] Redmine. *Redmine.org*. [Accessed April 15, 2015]. 2015. URL: http://www.redmine.org/.

[6] K.S. Rubin. *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley Signature Series (Cohn). Pearson Education, 2012. ISBN: 978-0-321-70037-7. URL: https://books.google.com/books?id=3vGEcOfCkdwC.

[7] Mountain Goat Software. *Product Backlog*. [Accessed April 24, 2015]. 2015. URL: https://www.mountaingoatsoftware.com/agile/scrum/product-backlog.

[8] Mountain Goat Software. *Scrum Team*. [Accessed April 16, 2015]. 2015. URL: https://www.mountaingoatsoftware.com/agile/scrum/team.