1 Methods

In this section we describe the development methods and the tools which are considered to be used in this project. A choice is given in both of these areas along with a reasoning for the choice.

1.1 Development Method

In this project we are four groups working together. For this collaboration to work properly it is important that we have a common understanding of the development method we are using. Different development methods are presented here and a single one is chosen to be used in this project.

1.1.1 Considered Methods

A common division of development methods is into traditional and agile categories. In general, traditional methods follow a structured plan, where every task is handled in a single unit and the result is not changed afterwards [**?** , sec. 2.7]. This is inspired by the construction industry, where these methods originate. Projects using agile methods are developed in iterations, where some part of the product is developed in every iteration [**?** , p. 25]. This should help cope with the changes that the end users or customer might pose during the development.

Some methods can have characteristics from both categories. The development methods considered in this project are: Extreme Programming, Scrum, and Waterfall. These are presented shortly in the following.

Extreme Programming Extreme Programming (XP) is an agile development method that consists of a series of 12 recommended core practices [**?** , p. 137]. These include but are not limited to: Frequent refactoring, pair programming, and whole team together. The core of XP is to find every practice that is considered good and taking it to the extreme, e.g. since code reviews are good, do them all the time through pair programming. Kent Beck, the creator of XP, states that in general all the practices of XP should be applied because they compensate and support each other [**?** , p. 156-157].

When using XP, the development is to progress in iterations of one to three weeks each. An iteration should not be planned until right before the start of it, where the onsite customer should help prioritize which features should be implemented in the given iteration and the programmers estimate the time to implement them. This process is called "Iteration planning game".

There are roles assigned to different people involved in the project [**?** , p. 145]. These roles are: Customer, programmer, tester, coach, tracker, consultant. All these roles are important, but Larman stresses the need for an onsite customer or at least an onsite customer proxies [**?** , p. 152-156].

Scrum Scrum is an agile development method that utilizes a number of iterations of development cycles. These development cycles are known as sprints. A sprint usually has a length of a few weeks. A sprint backlog is created prior to the sprint. This backlog consists of the features, ordered by priority, that should be implemented during the sprint. During the sprint the sprint backlog cannot be changed. If all the features in the sprint backlog are not implemented during the sprint they are moved to the product backlog, which is a list of the features that should be implemented in the future. In scrum there are different roles. There should be a product owner, whose task is to meet the costumer's and end users' interests. These interests should be formalized and prioritized in the product backlog. There should also be a scrum master. A scrum master should serve as a link between the development team and any external individuals. It is the scrum master's task to ensure that the development team is not disturbed. The development team is usually small, and their task is to design, analysis, and implement the features from the backlogs. Every day the scrum team holds a short meeting where they say what they have done since the preceding meeting, what they plan on doing the present day, and any problems they are having.

A variant of Scrum for more teams is called Scrum of Scrums or Large Scrum, where there are one complete team consisting of several smaller sub-teams, that will hold Scrum of Scrum meetings during each sprint to synchronize their work.

Waterfall A project following the Waterfall method is divided into a number of phases. The number of phases in a Waterfall development process is not exact. Depending on the person asked more or less phases may be suggested e.g. Marry Poppendieck is refering to six phases [**?**]. Waterfall is strictly traditional, since it in its pure form does not allow to move "backwards" in the life cycle. Where backwards refer to moving back into a phase once the phase has been completed. This suggests a big and heavy up front design plan that is to be followed when finished.

1.2 Choosing a Development Method

The following list shows the characteristics of this project that will be used to determine the development method which should be used.

1. 14 persons in four groups
2. Diverse target group
3. No onsite costumer
4. Hard deadline
5. Pass on project
6. Known framework and platform
7. Education environment

8. Not full-time development

9. No manager/Project owner

10. No shared working room

We will use a development approach similar to Scrum of Scrums. The reason for this is three fold. First of all we have a diverse target group (point 2), that may make a big up front analysis and design difficult. This leads us to choose an agile method due to "I know it when I see it" (IKIWISI).

Secondly we are 14 group members divided into four groups (as point 1 states) which is not handled very well in other agile methods such as XP, which dictates that all the developers should be in the same room. This is not the case for us as point 10 states.

The third reason is that we have a hard deadline (point 4), which means that we have to hand in our project at a specific date. Scrum of Scrums suggests that iterations (sprints) are time-boxed, which is ideal for us since we can cut less important features instead of missing the deadline. This is also supported by point 5, because the end product is a working release although some features may have been cut. The features cut may then be suggested to the group which is to take up this project next year.

1.2.1 Refining SCRUM

As point 9 and 3 states, we have neither a project manager nor an available on-site customer. We will handle the missing on-site customer by having shorter iterations and contacting the customers whenever an iteration is over. Scrum of Scrums dictates that there should be a Scrum master in each subgroup. Since none of us has used Scrum before, none of us are qualified to be Scrum master. We are in an educational situation (see point 7) so we will strive to allow every member to try to be Scrum master for a shorter time period. This may not be ideal, but we consider it to be more important that every member of the subgroups tries to have the responsibility of a Scrum master than having only one member trying it and learning it well.

1.3 Tools

A series of tools are used in creation of this project. These tools are briefly described in this section. The tools that needs to be chosen must indeed complete some task, the tasks that needs a tool are: Version control, bug tracking, code documentation, and testing.

1.3.1 Version Control

All of us have been using subversion (SVN) in previous projects as the version controlling system. SVN is a centralized solution [**?**] with a single repository that the group members can update from and commit changes to. This project does, however, differ from previous project with respect to the organization of groups; we are not one group of x individuals, but rather one group consisting of some smaller subgroups, again consisting of individual persons. This has lead us to choose a distributed solution rather than a centralized one.

The solutions considered are the distributed systems Git and Mercurial (Hg). These systems are quite similar and the essential difference is that Hg is simpler than Git. A few of us have been using Hg and none of us have used Git, which leads us to choose Hg, such that we a little knowledge of the chosen system.

1.3.2 Bug Tracking

We need to have some way of communicating and tracking the defects or "bugs" that we will run into during our project. The must important requirement for the tool we will use is that it should be able to track the bugs and make them easily available to the team that will continue on this project next year. The tools that we are considering to use for bug tracking are Bugzilla [**?**] and Eventum [**?**]. These tools are very similar in their sets of features. Since we have used Bugzilla as part of the course Test and Verification, it will save us the overhead of having to learn a new tool by select that over Eventum. Furthermore, development of Eventum seem to be discontinued since the start of 2009 [**?**], which means that any defects that may be in the system are very unlikely to be fixed. In conclusion we choose to use Bugzilla to track our bugs.

1.3.3 Code Documentation

We want our documentation of our code to be as close to the code as possible. In particular we do not want to go back and forth between the source code and some other program when we are writing new functionality and documenting it. We want to use a tool like PHPDocumentor [**?**] or PHPXref [**?**] to handle our documentation, since both of these reads comments in the source code and use it as documentation. This gives us the ability to write code and documentation in the same file, which is what we want. The syntax that both of these tools use is the same, which means that as long as we adhere to this syntax we can use either tool to generate documentation.

The difference between the two tools is that PHPDocumentor focus on giving a more diverse set of final documents (different HTML and PDF templates), where PHPXref focus more on the reference between classes, function, etc. We use both tools such that we and the group that is to take up this project next year can choose the documentation they prefer.

1.3.4 Testing

The test tool that we want to use should be easily adapted by us, which leads us to look at tools using PHP, since that is the language we will be using to develop this project. We also want the tests to survive after our work on the project ends, in particular the groups that is to take up this project next year should be able to use our test cases again. We will be using the build in testing framework of Moodle [**?**], which is based on SimpleTest [**?**]. This should ensure that the test cases can be used next year as well and perhaps be part of the final product, should it ever be realeased to the public.

References

[] Unit test api. URL `http://docs.moodle.org/dev/Unit_test_API`. Last viewed: 7/3-2012.

[] Simpletest: Unit testing for php. URL `http://www.simpletest.org/`. Last viewed: 7/3-2012.

[] Apache subversion. URL `http://subversion.apache.org/`. Last viewed: 6/3-2012.

[] Eventum issue / bug tracking system, 2012. URL `http://dev.mysql.com/downloads/other/eventum/`. Last viewed: 13/3-2012.

[] Eventum issue / bug tracking system, 2012. URL `http://dev.mysql.com/downloads/other/eventum/features.html`. Last viewed: 12/3-2012.

[] Individual bugzilla.org contributors. Bugzilla: Features, April 2011. URL `http://www.bugzilla.org/features/`. Last viewed: 12/3-2012.

[] The PHP Group. Package information: Phpdocumentor, 2012. URL `http://pear.php.net/package/PhpDocumentor/`. Last viewed: 6/3-2012.

[] Craig Larman. *Agile & Iterative Development*. ADDISON WESLEY, first edition, 2004. ISBN: 0-13-111155-8.

[] Mary Poppendieck. A rational design process its time to stop faking it, April 2000. URL `http://www.leanessays.com/2010/11/rational-design-process-its-time-to.html`. Last viewed: 6/3-2012.

[] Gareth Watts. Phpxref, December 2010. URL `http://phpxref.sourceforge.net/`. Last viewed: 6/3-2012.