Team Reflection

Our group had six members and so initially meetings were decided to be held after the lecture on Monday as the majority of members would be able to attend and members who did not attend the lectures were available after the lecture time slot so turned out to be extremely convenient for all members and resulted in a high attendance rate to the meetings.

We decided to have an additional meeting later in the week to increase our contact with each and it was held on Thursdays, generally online through Google Hangouts. These helped to allow team members to address issues to the whole group, which at the very start was highly effective to explain issues such as the infrastructure and technology we’d be using as well as any misconceptions on what user stories meant.

The whole team utilised Facebook extensively throughout the iterations which proved helpful in bringing up issues and asking questions in-between meetings and for topics that are not significant enough to be on the agenda for the Monday or Thursday meetings.

Although, communication between team members was not too bad, communication between the team and the customer could have been better. In terms of the functionality of the product and which user stories should be revised or re-prioritised the communication with the customer was fine, the relevant questions were asked during each meeting with the customer. However, when a team member could not attend a meeting the customer was generally not notified until the actual meeting which was an example of where the communication between the team and the customer could have been better.

The system our team used to distribute user stories through Jira was a more liberal one. We prioritised user stories and had the highest priority user stories to be completed in the earlier iterations and lower priority user stories to be completed in the later iterations or not at all i.e priority by ‘Must Haves’, ‘Should Haves’, ‘Could Haves’ and ‘Won’t Haves’. Each iteration everyone would choose user stories they thought they could complete, first come first served. This method worked somewhat, as we have a majority of the functionality that was promised implemented.

After the first iteration we observed from the burndown chart that the user story points were significantly underestimated because we did not account for the new technology learning curves and overhead such as setting up relevant IDEs and other infrastructures such as the ftp server for server-side scripts. This was promptly accounted for in the future iterations by using our velocity as a benchmark and prioritising user stories per iteration using our current velocity (the velocity at the end of each iteration) as a maximum. This proved to be extremely effective and helped prevent promising more user stories in an iteration than we could reasonably deliver.

The day before the final release was due a team member had overwritten the server-side script file with an older version. Fortunately, another team member had uploaded a later version (but not the latest) to the Stash repository and was able to manually update it to the latest version with help from another member who had an even later version on his local repository. This was a good lesson to teach the importance of the usage of a repository, as well as using each team member as a form of version control and regularly pulling and pushing to the repository so that as many members as possible have the latest working version.

Adaptation of XP Principles

* Regular stand up meetings were useful but not practical in our situation with everyone having different schedules.
* Facebook allowed more flexibility with meetings.
* Some meetings did not have everyone in attendance. Low understanding in underlying development concepts meant user stories had varying levels of dependencies not accounted for.
* Informal code reviews were undertaken in the earlier weeks
* Formal code reviews were written by most team members for the last two iterations
* Due to some work types not being on Jira such as documentation, not all tasks could be tracked.
* Continuous integration took a while to set up because of higher focus on initial functionality and learning.
* Testing of highly interactive activities was much harder which lead to fewer tests and lower code coverage.
* User stories were well thought out with minimal changes or modifications from the customer.
* Acceptance testing was done on all tasks before being marked as complete by making sure the acceptance criteria were met.
* Spikes in our burndown charts resulted from underestimating several user stories in each iteration
* Building by Jenkins with every commit allowed us to see if someone had committed code with compile errors.
* Code reviews helped with making sure refactoring was occurring.
* Test driven development for the most part was hard due to lack of background knowledge, in some isolated classes such as the minigame TDD worked well.
* Planning to “catch up later” seemed to occur with a few people.

Low coupling and high cohesion were enabled by creating separate classes for different tasks (such as DB connections) which also improved unit testing ease.

Customer Engagement

* Customer encouraged us to revise a particular User Story involving recording audio files, we ended up dropping this particular story as a result.
* Consulted with the customer early on to determine whether or not a ‘Patient Registration’ module was necessary for the prototype, it was not deemed to be so and we pursued it no further.
* Consulted with the customer about the priority of having a professional UI, we were encouraged to focus on functionality above all else and so we shifted our priority, resulting in a basic looking Prototype but with some advanced functionality.
* Because of some previous features not being as expected, the customer decided to postpone the development of a mini-game while other issues were sorted.
* Application notifications were prioritised after the customer expressed interest.
* Release functionality was discussed just prior to releases when we had a good idea on what was possible.