Cross-Team Communication in Application Development

Benjamin Denham
Manukau Institute of Technology
ben.denham@gmail.com

Rosemarie Tomlinson (supervisor)
Manukau Institute of Technology
Rosemarie.Tomlinson@manukau.ac.nz

ABSTRACT

This paper and accompanying poster are a summation of lessons learned on the nature of cross-team communication over the course of a software application development project involving distributed teams.

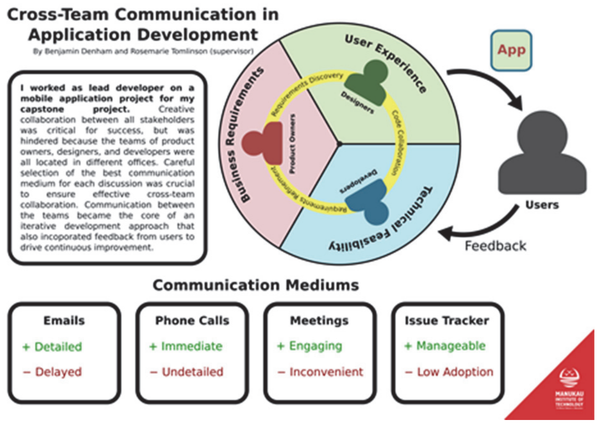
Keywords: software development, mobile application, communication, collaboration, distributed team project

1. INTRODUCTION

Collaboration between stakeholders is a key factor in the success of any software development project. However, effective collaboration can become difficult when teams are distributed geographically.

I gained first-hand experience of this kind of difficulty while working as lead developer on a mobile application for my capstone industry project. The product owners, designers, and developers all worked for different companies from separate offices. The challenge was to collaborate effectively and maintain efficient communication channels between the distributed teams.

This paper and accompanying poster describe the nature of the collaboration and the various communication mediums that were used; comparing them according to their strengths and weaknesses.



I worked as lead developer on a mobile application project for my capstone project. Creative collaboration between all stakeholders was critical for success, but was hindered because the teams of product owners, designers, and developers were all located in different offices. Careful selection of the best communication medium for each discussion was crucial to ensure effective cross-team collaboration. Communication between the teams became the core of an iterative development approach that also incoporated feedback from users to drive continuous improvement.

This poster appeared at the 6th annual conference of Computing and Information Technology Research and Education New Zealand (CITRENZ2015) and the 28th Annual Conference of the National Advisory Committee on Computing Qualifications, Queenstown, New Zealand, October 6-9, 2015. Michael Verhaart, Amit Sarkar, Rosemarie Tomlinson and Emre Erturk (Eds).

2. COLLABORATIVE DEVELOPMENT

There were three core teams that contributed to the mobile application project.

- The product owners - employees of the organisation that the application was being built for. Their focus was on ensuring the delivered app met the business requirements of the organisation.
- The designers - worked for a design agency that had been contracted to develop the look and feel of the application. Their focus was on ensuring the application delivered a clean and useful experience for users.
- The developers (including myself) - worked for another agency that had been contracted to build the application itself, and connect it to other systems via RESTful APIs. Their focus was on determining what features were technically feasible for the application, and then implementing the required functionality.

Collaboration between each pair of teams tended to have a consistent purpose throughout the development process. The product owners and designers worked together to determine what functionality the application should have, and how it should behave. The developers would then work with the product owners to further refine those requirements by asking detailed questions about how particular usage scenarios should be handled, and also inform them of any technical limitations that would prevent certain functionality from being implemented (such as appropriate APIs not existing in underlying systems). Finally, developers and designers would work together on a shared codebase to integrate business logic with application layout and styling.

The open-source Git version control system proved invaluable for managing this code collaboration. It allowed designers and developers to work completely independently on separate “branches”, and then merge their work together later on. It also enabled the coders to see who was responsible for each line of code using the git-blame tool (Git Documentation, n.d.). This meant that they could seek assistance from whoever was responsible for the original work, and discuss any planned alterations with them.

As part of the iterative development process that was practiced in this project, new versions of the application were frequently delivered to users for testing. The users would then deliver feedback to the product owners regarding the usefulness and user experience of the application. This feedback would then drive new features and improvements to be developed and added in the next version of the application.

3. COMMUNICATION MEDIUMS

We utilised a variety of communication mediums and found that each medium had its own unique strengths and weaknesses which made it useful in certain situations, but ineffective in others.

3.1 Emails

Emails were the primary method of communication. We sent detailed questions to all involved parties. Recipients could take the time needed to gather all relevant information and craft an informed response.

Some team members would not respond to emails promptly or forget about an email they had received, creating delays of up to several days between questions and responses. While email conversations were excellent for discussing all relevant details to inform decisions, coming to a decision could take days or even weeks.

When faced with many separate questions recipients would tend to respond to only the first question presented in an email.

With large lists of recipients, it became easy to accidentally leave someone out which led to further confusion and communication breakdown.

Overall, emails were a useful medium for technical discussions, but at times certain aspects of their nature made communication highly inefficient.

3.2 Phone calls

We had a conference call each day with everyone involved in the project. This acted as a kind of daily stand-up/scrum meeting (Cohn, M., n.d.). Each person involved explained what they were working on and noted any problems they were facing or anything they were waiting for. These conference calls also became a good time to follow up on any emails that had been sent recently.

Additional phone calls were also made throughout the day when confusion arose in an email conversation, or when a recipient was taking too long to respond to an important email.

One major benefit of these phone conversations over emails was the ability to receive an immediate.

If someone did not have access to all of the information they needed to make a decision at the time of the phone call, then the call became less useful, and the discussion had to revert back to emails.

It was difficult to fully explain all technical details relevant to a particular problem through a phone call.

Another problem with phone calls was that there was no inherent record of the conversation, so it was important to note down the main points that came out of any conversation.

Over the course of the project, phone calls became an invaluable tool to make up for many of the problems inherent in email communication.

3.3 Meetings

In-person meetings were very infrequent because it was difficult and inconvenient to have all of the teams to meet together.

Meetings were used to discuss which features should be developed over the following few weeks, or to review a large number of recently developed features in the application.

We would also schedule meetings to make technical decisions that were of critical importance, such as how the application should handle API versioning and deprecation.

The best meetings were those where all teams attended, as problems could be analysed from all perspectives in order to come up with the best solutions.

Face-to-face meetings were probably the most productive methods of communication and collaboration used in the project, but the inconvenience of having the distributed teams meet together meant they could only be used sparingly.

3.4 Issue Tracking

We also utilised the Redmine (http://www.redmine.org/) issue tracking and project management tool on the project. We used it to maintain a backlog of features to be developed, along with their status. This allowed us to keep track of how much progress we were making, and give us an idea of which features could be developed before the next scheduled release.

Because of the agile nature of the project, the scope of tasks to be completed was constantly changing. Some tasks would be removed from the backlog while others would be added.

The interface of Redmine did not make it convenient to update tasks. The result of this was that the backlog on Redmine would become out of date, and not give an accurate picture of the state of the project. The developers had meetings to go through each task and update it, but found that using a task-list on a whiteboard was more practical most of the time.

Redmine allows you to post messages in a discussion thread associated with each task. We used it as a replacement for email discussions around tasks. This was useful, because it meant that all discussions related to a particular task were kept in a single, manageable thread instead of in many separate emails. It also meant that there was never any risk of someone not being included in the discussion because everyone had access to all of the discussions.

We found that it was difficult to get all of the team members to adopt usage of Redmine. Because of this, messages posted to a task in Redmine would often be ignored for several days. We found that we had to resort to email to communicate with team members who were not using Redmine frequently enough. Redmine became an inconvenient overhead, and we largely resorted back to using emails for discussions. Managing tasks in a tool like Redmine can solve many of the issues associated with email communication, but it requires all team members to fully adopt the tool.

4. CONCLUSION

Like any project involving distributed teams, our project faced many challenges in collaboration and communication. These issues were mostly mitigated by effective use of available communication mediums.

I believe this project highlighted the importance of effective communication in ensuring the success of a software development project. Setting up and maintaining communication channels that meet the needs of everyone involved should be prioritised in all software development projects. Furthermore, all team members should be on the lookout for problems with communication so that they can be addressed as quickly as possible.

5. REFERENCES

Cohn, M. (n.d.). *Daily Scrum Meeting.* Retrieved from http://www.mountaingoatsoftware.com/agile/scrum/daily-scrum

Git Documentation. (n.d.). *git-blame Documentation.* Retrieved September 13, 2015 from http://git-scm.com/docs/git-blame