Team Project 3 Individual Report
A social network for project management

Tom Wallis
Matriculation No.: 2025138

February 6, 2015

1 Preliminaries

A project management tool or suite is a program or set of programs which allow a team of developers, managers, and customers to understand the state of a project. These tools tend to be used to manage *tickets*.(The act of managing tickets is referred to as *ticket management*.) *Tickets* are notices that members of a development team can set up, discuss code on, and see information about a problem such as its severity and who is responsible for fixing this. *Issues* and *bugs* would be found recorded in *tickets*. *Tickets* are also used to see what development has been done on a project in the past.

Examples of project management tools used today are *Trac*, *Jira*, *Asana* and *Trello*. Others, such as *Codebook* by Microsoft, have attempted to produce a system where social networking allowed for collaboration on bug reporting. However, Codebook saw little adoption and was not ideally user-friendly, using graph theory to draw connections between people and bugs.

In a software development team, one finds that there exists a set of *developers* who have their work managed by *Quality Assurance Managers*. This whole system is overlooked and monitored by a *Project Manager*.

When one develops code, to keep working and non-working versions maintained side-by-side one employs *branches*. These *branches* contain different versions of code which are being worked on and maintained by different sets of developers. Often, quites of software that manage *branches* can manage *issue tracking* in the form of *ticket management*. An example of a tool that does this is Github. Often, a development team will use the most stable branch of their project while developing their new version; this process is called *dog-fooding*.

2 Motivation

When writing a software project as a team, it can be difficult to manage issues and bugs. This is generally done with a project management suite such as Trac, in the form of tickets. However, it is often found that programmers discuss these tickets in an informal environment and, once a conversation is had detailing the issue, this work must be carried out a second time in the creation and comments section of a ticket.

This is a problem for two reasons: firstly, it duplicates work and wastes time. Secondly, project management tools can be cumbersome to work, making their integration into people's work schedule difficult.

This problem is clearly one worth solving, and is one we are faced with daily as software developers working in a team.

To address this, we will build a conversation-oriented system which allows for ease of use similar to a standard social networking site. However, using metrics obtained from conversation data and metadata generated and inputted by users, developers will be able to track and manage bugs and issues in software development projects.

3 Background

There are several related works we have had to build our ideas upon in the design stage of this project.

- Asana
  Asana is a system which attempts to remove the need for email to update people regarding news they are to be aware of: due dates, tasks, comments on their work etc. Asana was founded by Facebook executives and developers initially working on improving productivity at Facebook. Asana is a useful tool, but aims to pull together many different pieces of information into one central hub – the idea powering the product is to *kill email*.
  Because we are looking to collect only information and meta-information about tickets, Asana does not solve the problem noticed in the programming world.

- Codebook
  Codebook is a social networking oriented way to manage code repositories and the people who maintain them. Codebook creates graphs and connections between people and project management artifacts like tickets.
  Codebook doesn't quite solve the problem we are faced with, however. This is because, while the social network Codebook creates allows managers, developers, and customers to link together, the graphs it creates are complex and do not necessarily permit easy communication. While networking people and code together is useful, the main concern here is that programmers will communicate using channels other than their project management tools.

- Trello
  Trello is a web-based tool based on the KanBan system of managing lists. It can be used to track and archive progress at various specific points in a project's development cycle.
  However, while Trello is useful, it can also add to the clutter we are seeking to prevent. Trello does not actively track tickets and also does not allow for "chats" – instead, it manages lists of arbitrary things and provides a commenting system and history of recent actions on a board of cards.
  As an aside, it should be noted that our team uses Trello alongside Facebook to manage our own tasks, as a to-do list.

- Slack
  While only a young startup, *slack. com* is of note due to the similarities between this project and their product. `Slack.com` provides a conversation-based way of managing projects, and shares most of this project's design goals. Crucially however, slack.com is geared toward general project management and is not heavily tailored to software development projects. Additionally, `slack.com` has less of a focus on metadata than this project. `Slack.com` has become the focus of much attention, and has routinely found millions of dollars in funding since its release two years ago.

As can be seen, there have been many attempts to solve the problem identified in this introduction's *Motivation* section. However, none of them come at the problem at an angle which allows for all conversation and project-management to be done in one place. Therefore, a new system must be devised.

4 Aims

This project attempts to cumulate the functionality of Asana, Codebook, and Trello into one web-based package where project managers, developers, and customers can discuss a project in a social-networking oriented way. Using these conversations, this project should then develop *tickets* as the conversations being had with calculable and attachable metadata.

The project should be able to socially network at least developers, their quality assurance managers, and their project managers, and should be able to manage multiple projects. This is functionality also found in Codebook. It should be able to process metadata like Asana can, and allow users to keep track of issues and

bugs like Trello enables with its lists. However, the project should *also* do this while fixing the flaw found with all three aforementioned solutions: it should do so in a way centered around conversations.

With appropriate user interface design, using conversatons as tickets should also solve the three issues laid out in the *Motivation* section:

1. The project allows for discussion of the issues and bugs within their project management suite.
2. Because discussion occurs directly within a ticket, there is no need to duplicate work as there would be creating tickets on the basis of Skype or Facebook conversations.
3. An easy-to-use user interface would allow for easy covnersation management, meaning that the tool would be able to implicitly manage "tickets" through conversation and metadata management, which is diffuclt to do with tools such as Trac.

5 Progress

Firstly, meetings were held with a project supervisor to determine exactly what the requirements of the project were. Our development team were given a choice to integrate conversation and social networking capabilities into an existing ticket management system such as Jira via a plugin, or to create our own solution, potentially as a website hosted online. We decided to create our own project, as it would allow us to resist restriction in the form of another project's setup and style of working.

Currently, the project is nearly in a useable state for communicating as a chat platform. Chats exist and sync between different browser sessions. As time goes on, profiles will become more useful for users and networking will be implemented. Currently, user types – Developers, Q.A. Managers, Project Managers – exist, but are being merged into a beta branch of the code.

The project is being written in Django on the backend. This was done for a number of reasons:

- Django is a Python backend, and Python is a relatively simple langauge to program in.
- Django's template system is powerful and provides all of the functionality we need.
- Django provides a user model with permissions and groups we can utilise to ease the social networking aspect of the project.
- Django has copious documentation and an active developer community. This was particularly important, as the majority of our team had no prior experience in web development.
- Django's model framework is both powerful and intuitive, and ER diagrams are easy to develop as Django code.

Once requirements had been solicited from the project supervisor, project setup was initiated. Most of the work performed for the first month was simply setting up the framework we would create the rest of the project in. Once this was done, actual pages of the website – chats, project pages, permission settings, and so on – were added to the project. Additionally, admin pages had to be written and configured so that the project management tool could itself be managed by some superuser. Naturally, now that much of the backend of the code has been written, future features will be far easier to implement.

The code was developed primarily in Jetbrains' PyCharm IDE, and is being maintained on a Github page. The team meet at least once per week to discuss current issues (which are being tracked on Trello) and to catch up on work each other have done. The project currently has a master branch being hosted on Heroku.

6 Conclusion

To conclude, the project's development has run smoothly so far. Ample progress has been made, and we feel we are sticking to the requirements solicited with the project supervisor at the beginning of the project (and in subsequent weekly meetings also).

An interesting observation is that the utility of the program has become evident during its own development. While bugs are discussed on Facebook Messenger, detailed on Trello, and written on Github, there can sometimes be confusion as to who is developing various parts of the program, and there are often problems with information kept on Trello being out-of-date. While this is partly due to the management skills of the team, it also has to do with the tools available. A tool such as the one we are developing would be very useful to us, and this knowledge makes it easier to develop something useful. A mantra we can employ would be that *if we would use it for managing this project, other people will want to too.*

As a result, we are eager to begin dog-fooding and use the tool ourselves while it is in development.