Stony Brook University

Rapid Mobile Software Development for Field Data Collection Applications

A Thesis Presented

by

Sushal Penugonda

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Master of Science

in

Computer Engineering

Stony Brook University

May 2016

Stony Brook University

The Graduate School

Sushal Penugonda

We, the thesis committee for the above candidate for the

Master of Science degree, hereby recommend

acceptance of this thesis.

Fan Ye
Assistant Professor of Electrical and Computer Engineering

Shan Lin
Assistant Professor of Electrical & Computer Engineering

This thesis is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

Abstract of the Thesis

Rapid Mobile Software Development for Field Data Collection Applications

by

Sushal Penugonda

Master of Science

in

Computer Engineering

Stony Brook University

2016

The purpose of this paper is to describe a template system that can be used to design and deploy mobile applications for the purposes of collecting large volumes of data. The implementation referred to in this paper pertains to phone sensor data collected for research studies in the field of indoor localization conducted at Stony Brook University, however many of the concepts can be generalized to other applications. The system is described through team structure, coordination methods, progress evaluation methods, and system integration techniques. Coordination methods involve a custom implementation of agile development, a version control software, knowledge transfer methods used among the team members and project transfer methods. It is important to note that this type of system is developed around a team of students who have limited time to devote to the project and also a limited but diverse skill set.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my advisor, Fan Ye for the mentorship and patience. I would also like to thank everyone on my team for their dedication and enthusiasm to our project. I am grateful for everyone's patience as we tried numerous techniques to organize our efforts and for their honest suggestions for this thesis.

1. Introduction

The project we will be discussing was the task to develop two applications, one for android and another for iOS platforms. The purpose of this app is to collect inertial sensor data and relay collected data to a centralized file store. There had been an android and iOS app that a few students had started development on already, and I was handed over the task of developing them in the beginning of Fall 2015 semester. As the semester proceeded, our team consisted of three undergraduate students and three master students including myself. I took the place of team lead and we started developing without a formal structure. Every week our whole team met with the professor to show him our progress and it was then that we faced many issues. Over time, the android and iOS dev team deviated in the design of the app, and as a result the morale of the team diminished which in turn resulted in affecting the performance for the upcoming week.

Some of the issues we were facing was finding a common time to meet on a weekly basis, waiting for other students to finish their task before starting yours, having the same features and user interface elements and functionality on both platforms. Balancing the work among all the team members, avoiding code developed by many people to be resulted in lava code. Since the skill set of the team was not proficient in developing for mobile platforms, we were also met with a very high learning curve, this paper also goes over some ways one could minimize the effects of research spikes in the development process.

In this project, the project stakeholders are the people leading the localization research who will be using the app directly. The stakeholders would like to create consumer grade applications of these apps to open the data collection aspect to the general public.

2. Team Structure

Agile system used

Agile was the workflow of choice because of the flexibility it offers over other methodologies and the quick turnover rate of a usable product. Although in our case the first few releases of apps did not have consumers or field testers.

A good agile project consists of three main pillars:

- a task tracking system,
- regular scrum meetings, and team meetings.
- Robust development tools

Development tools are looked at in greater depth in Chapter 3. Cordination methods. During the course of the project, we used two systems: Google sheets and Trello for task management, each of these are covered in depth in the later sections. As the project lead I was the sole person meeting with the stake holders to avoid confusion and to have a slim agenda.

The main purpose of a scrum meeting is to present to the stakeholder the work that had been performed in the previous story, get their feedback and incorporate it in the stories ahead without going over the technical aspects in detail. Soon after the scrum, Our team uses a group chat application called groupme and we have a discussion with an objective of assigning tasks to all the developers. It is very important to give stakeholders a clear cut picuture of what you are proposing and so on multiple occations we used mockup UI to demonstrate what functionality we were aiming for.

On a side note the UI was designed in opensource image editing software called GIMP and was presented using a free online tool called Invision. One could also use Lucidchart to make wireframe UI mockups. Tools like Invision allows one to rapidly develop a working UI just from images. Below are the screen shots of the initial app proposal:

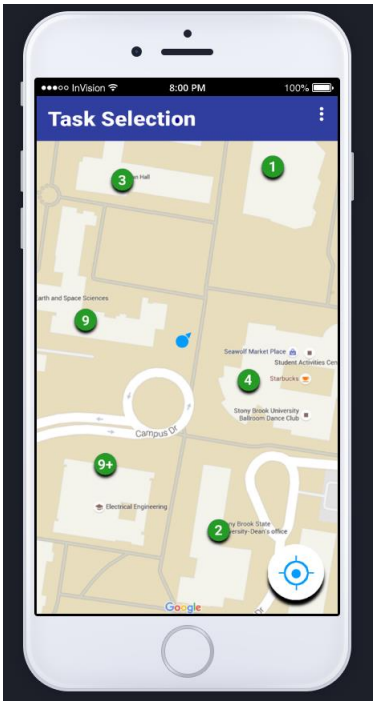


Figure 1. Invision Mockup screen 1

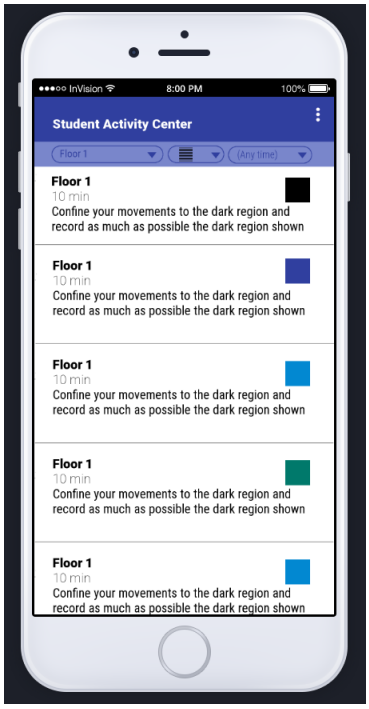


Figure 2. Invision Mockup screen 2

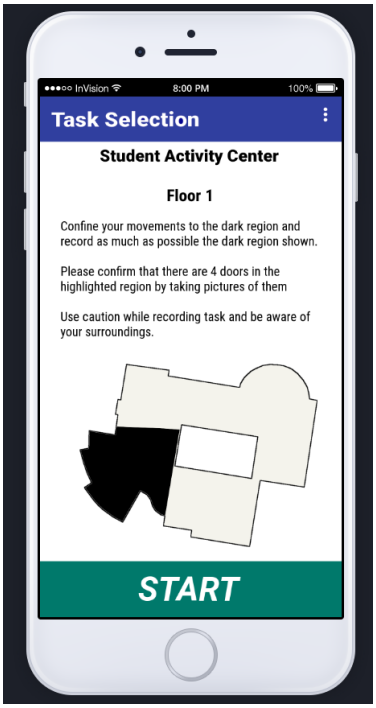


Figure 3. Invision Mockup screen 3

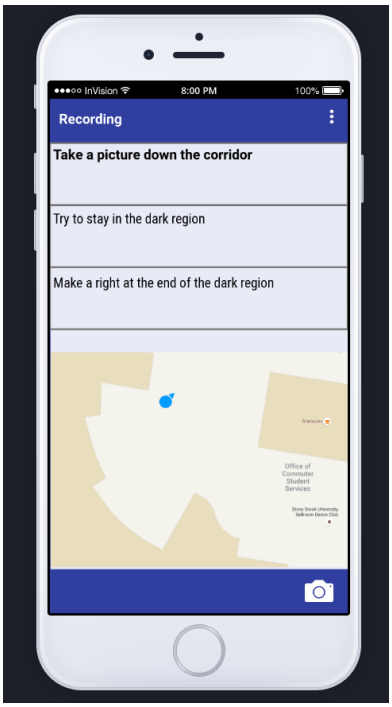


Figure 4. Invision Mockup screen 4

Here is a link to the interactactable version of this UI : https://invis.io/NW60AC25H#/134891088_Map_View. The set of screens after the final implementation are shown by the set of figures below. Only screen three and four from the figure above have been implemented but have changed drastically from initial proposal. In cases like this the team lead needs to ensure that the mockups are kept updated so the developers can reference to them.

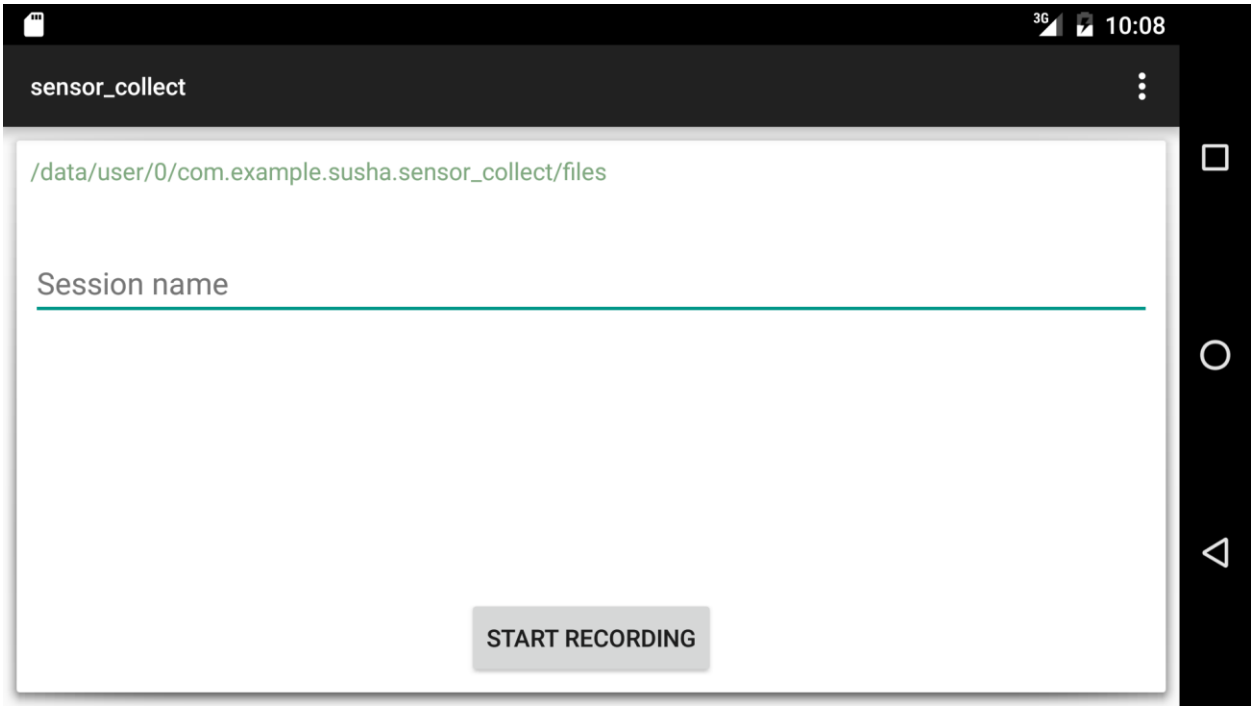


Figure 5. Android app screen 3

Figure 6. Android app screen 4

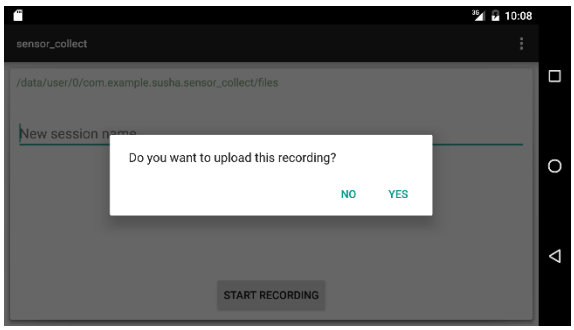


Figure 7. Android app screen 3, dialog 1

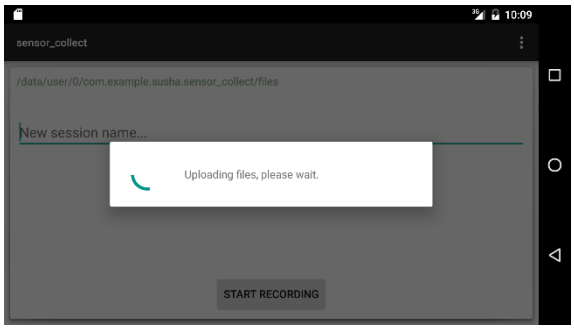


Figure 8. Android app screen 3, dialog 2

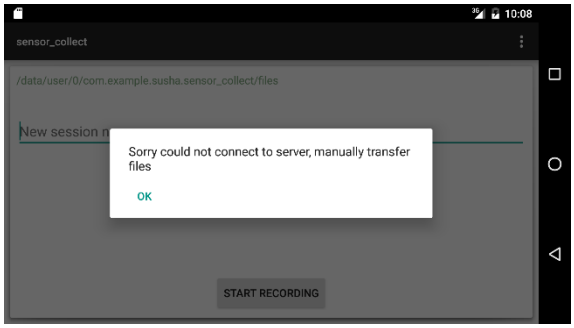


Figure 9. Android app screen 3, dialog 3

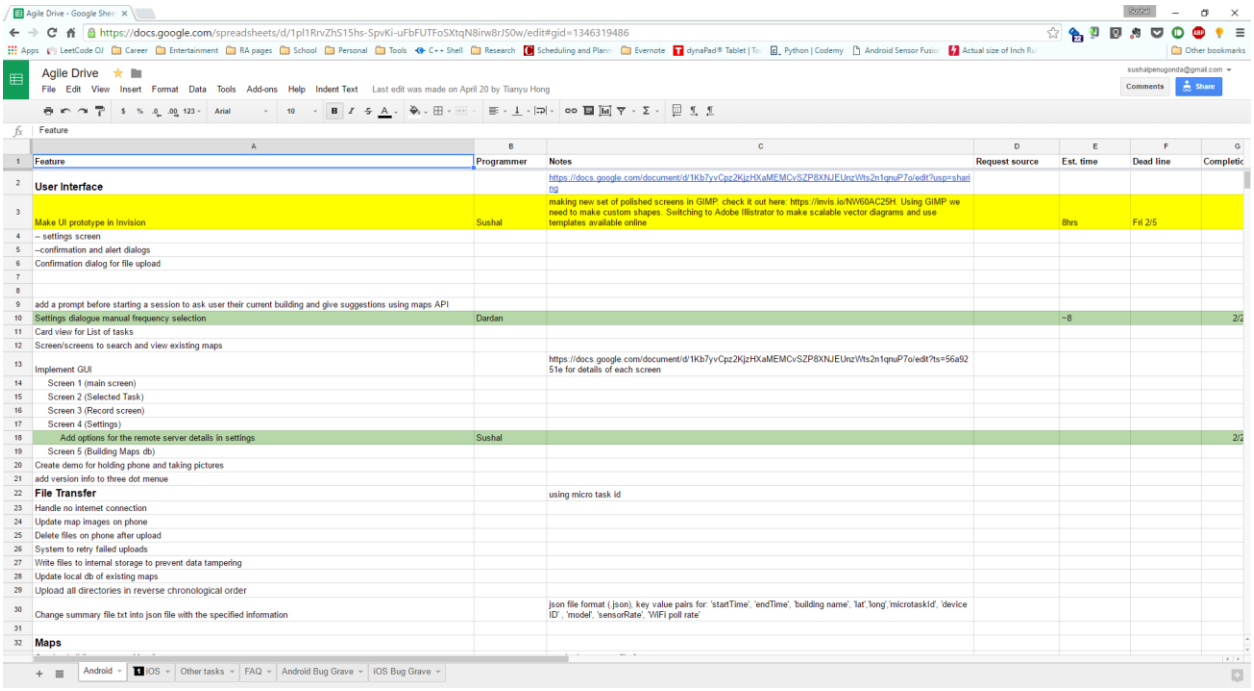
Story and Task Tracking

A story is a time period in which the team accomplishes a meaningful, tangible and presentable amount of work. In our case we chose a two-week time period as the duration of a Story. In each story, team members should be assigned at least one task. After completing their task they should be encouraged to help their teammates complete tasks or research for upcoming task. Our implementation of agile comprises of stories that are two weeks long, this is sufficient time for a team of students to devote 12-20 hours for a task.

For the beginning half of the project we were using google sheets – a web-based worksheet application. Ideally one would also want to track stories along with the lifecycle of tasks. Categories tracked in a task management tool are as follows:

- Feature related
  - Sensors
  - Maps
  - File Transfer
  - UI
- System related
  - INFO
  - Performance
  - Bugs
- Process related
  - In Dev
  - Dev Review/Testing
  - Completed
- Others

- Eg: Migrating to SVN



	Feature	Programmer	Notes	Request source	Est. time	Dead line	Completic
2	**User Interface**		https://docs.google.com/document/d/1Kb7yvCpz2KjzHXaMEMCvSZP8XNJEUnzWts2n1qnuP7o/edit?usp=sharing				
3	Make UI prototype in Invision	Sushal	making new set of polished screens in GIMP. check it out here: https://invis.io/NW60AC25H. Using GIMP we need to make custom shapes. Switching to Adobe Illistrator to make scalable vector diagrams and use templates available online		8hrs	Fri 2/5	
4	– settings screen						
5	–confirmation and alert dialogs						
6	Confirmation dialog for file upload						
7							
8							
9	add a prompt before starting a session to ask user their current building and give suggestions using maps API						
10	Settings dialogue manual frequency selection	Dardan			~8		2/2
11	Card view for List of tasks						
12	Screen/screens to search and view existing maps						
13	Implement GUI		https://docs.google.com/document/d/1Kb7yvCpz2KjzHXaMEMCvSZP8XNJEUnzWts2n1qnuP7o/edit?ts=56a9251e for details of each screen				
14	Screen 1 (main screen)						
15	Screen 2 (Selected Task)						
16	Screen 3 (Record screen)						
17	Screen 4 (Settings)						
18	Add options for the remote server details in settings	Sushal					2/2
19	Screen 5 (Building Maps db)						
20	Create demo for holding phone and taking pictures						
21	add version info to three dot menue						
22	**File Transfer**		using micro task id				
23	Handle no internet connection						
24	Update map images on phone						
25	Delete files on phone after upload						
26	System to retry failed uploads						
27	Write files to internal storage to prevent data tampering						
28	Update local db of existing maps						
29	Upload all directories in reverse chronological order						
30	Change summary file.txt into json file with the specified information		json file format (json), key value pairs for: 'startTime', 'endTime', 'building name', 'lat','long','microtaskId', 'device ID' , 'model', 'sensorRate', 'WIFI poll rate'				
31							
32	**Maps**						

Figure 10. Task Management Tool 1, Google sheets

Later we switched to Trello for task management because of its ease of access and a more user friendly, customizable design. Below are screen shots of Trello

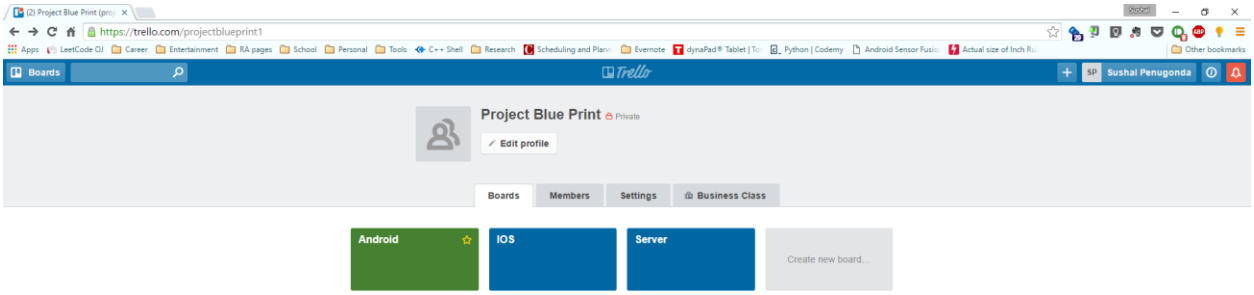


Figure 11. Task Management Tool2, Trello, screen1

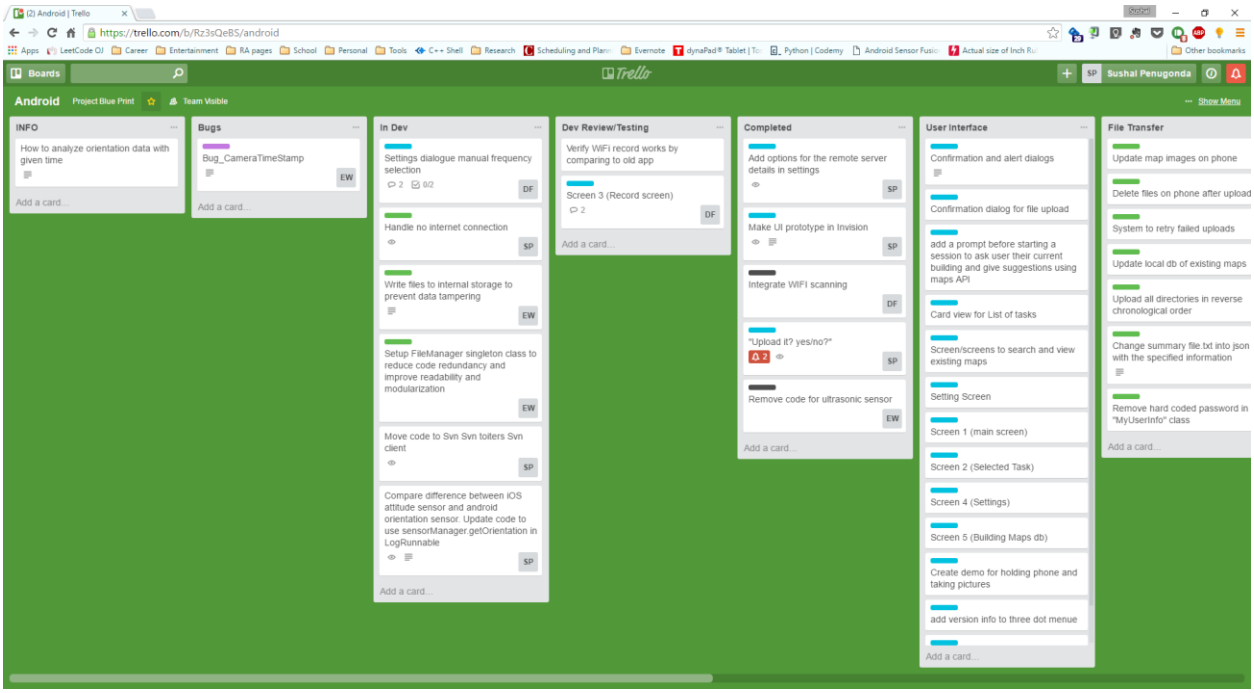


Figure 12. Task Management Tool2, Trello, screen2

Responsibilities and expectations

As outlined in the previous paragraphs, it is important to stress to new or prospective team-members the exact amount of hours they are expected to put in and give emphasis on the task oriented nature of the work. As such, this is one of the few expectations team-members need to abide by.

Before the beginning of the project it is recommended that the stakeholder meet with at least the team lead and discuss what each-others expectations are in **very specific** terms. Some examples of these expectations are:

- The stakeholder is expected to meet the team lead once, every week for at least 30 minutes to evaluate their progress and advise them on future tasks.
    - (On a side note, if such a requirement can't be met, then agile may not be the best choice)
- The team lead is expected to be transparent of the issues faced by the developers
- A total of 50 person-hours are expected to put in before December 31st
- A feature is expected to be implemented within 3 weeks of initial request

Ideally this initial discussion would be a written signed by both parties. As such, the team lead is expected to the best advocate for the rest of the developers. If not, another representative is expected to be appointed in addition to the lead. The same conversation is to be had between the developers/team-members and the team lead. The expectations contract would the first place to start when issues arise such as when a team-member is not completing their work on time or if the quality of work is poor. Hence, it is absolutely essential to have the contract as detailed as possible.

It is vital to bridge the gap between stakeholders and developers, as a means of checks and balances across the board. This can be accomplished by having a less formal scrum meeting once

every 2 or 3 stories where developers can present to the stakeholders the work they did. This time frame also for a significant amount of work to be accomplished. This meeting could focus on the system integration aspects of the project and help the stakeholder see it as a whole. If there are surprises in this meeting to either party, something serious needs to be addressed in the team dynamic of the project.

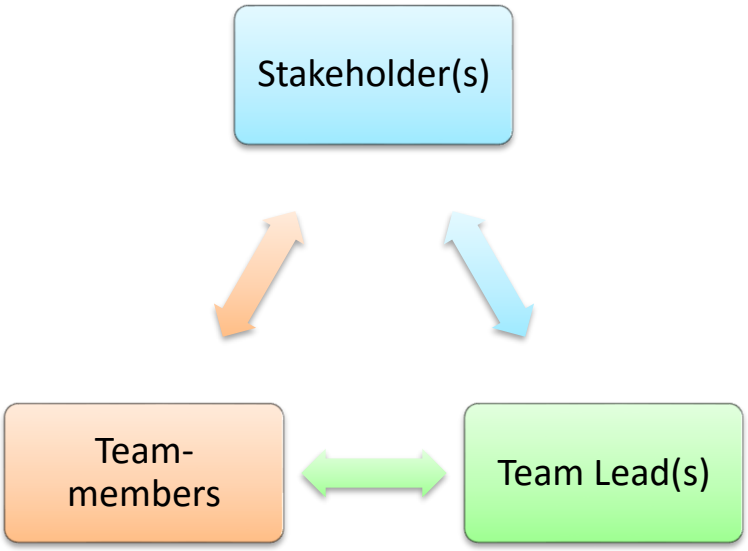


Figure 13. Team dynamic

3. Coordination methods

Version control software

Code review and commit comments. A centralized working repository works best for all the mini projects that belong the system. This makes it easier for the person merging the changes to access updates and for developers to use shared design languages/ resources to keep things consistent across the board.

Git flow and Integrating Git branches

The branching is inspired from git flow where in each the master branch contains the most stable version the code. There is a development branch which is used to clone new work spaces from by developers. Below is a figure that shows a good representation of Git Flow.

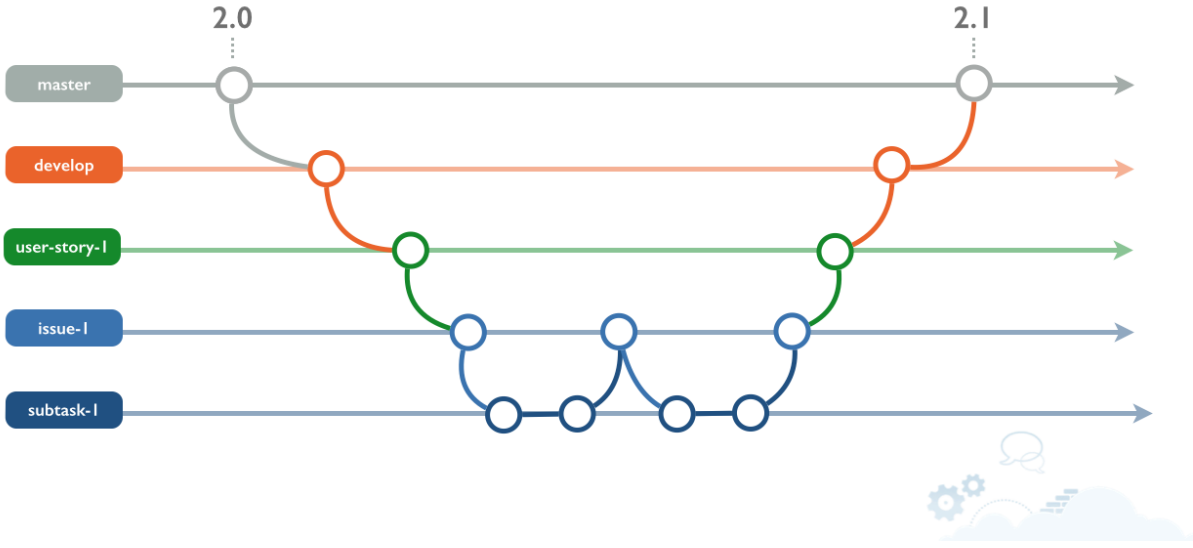


Figure 14. Git Flow

training new team members and team leads

A rigorous screening process is recommended for recruiting new members into the project to ensure the quality of resulting product. This would be done by the team – lead in close conjuncture with the stake holders.

Every new member who is a developer need to be trained in the way the team uses version control, source code walkthrough and environment setup. This task could be offloaded to a team-member who is most experienced or who is most likely to work with the new person.

4. System Integration Techniques

Role of lead

There can be multiple team leads to handle the work load of the project. Their roles other than that discussed in the sections above is to keep a close eye on merge requests from developers at the end of each story. They need to ensure that from the commits, the developer had accurately captured the delta (changes in code) for an associated task and send back the unqualified ones with comments to be updated. This is what could make or break the system.

System testing

In a system such as this it is hard to come up with an automated way to check data validity without processing it on the back end. Since in our case The stake holders were developing an algorithm that could take inertial phone data and map out an approximate floor plan. At the time of initially developing the apps, these back end algorithms were not finalized and so we went with manual checks as outlined by the following table:

Aspect of test	**Test description**	**Illustration**

Accelerometer	Slide the phone on a flat surface where the phone is stationary at the start and end points	Figure 15
Orientation	Move the phone along each of the three axis and back to its original position.	Figure 16
GPS	Use online services such as google maps to plot each of the location data points	NA

Table 1. Testing table

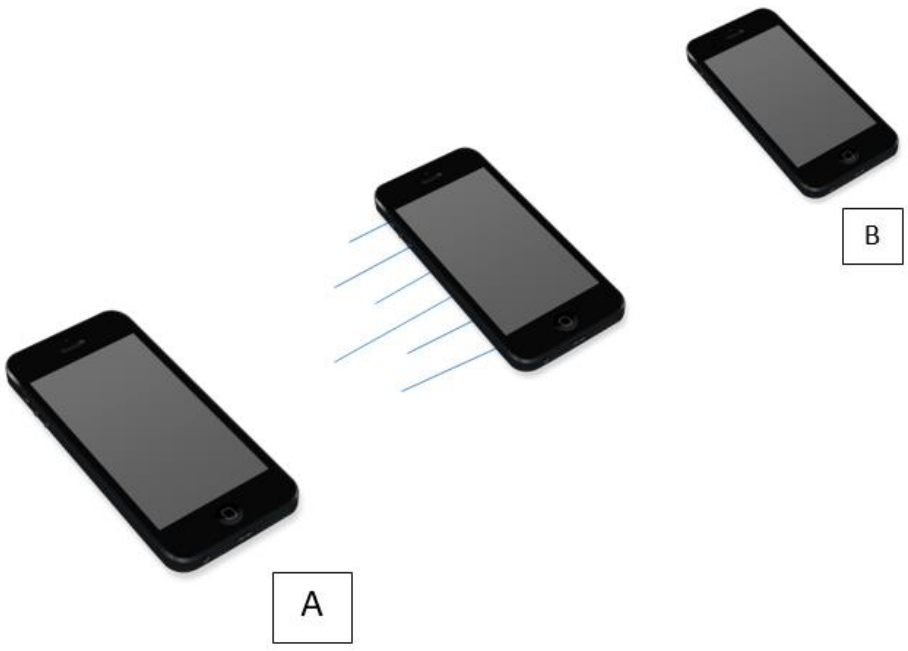


Figure 15. Accelerometer testing

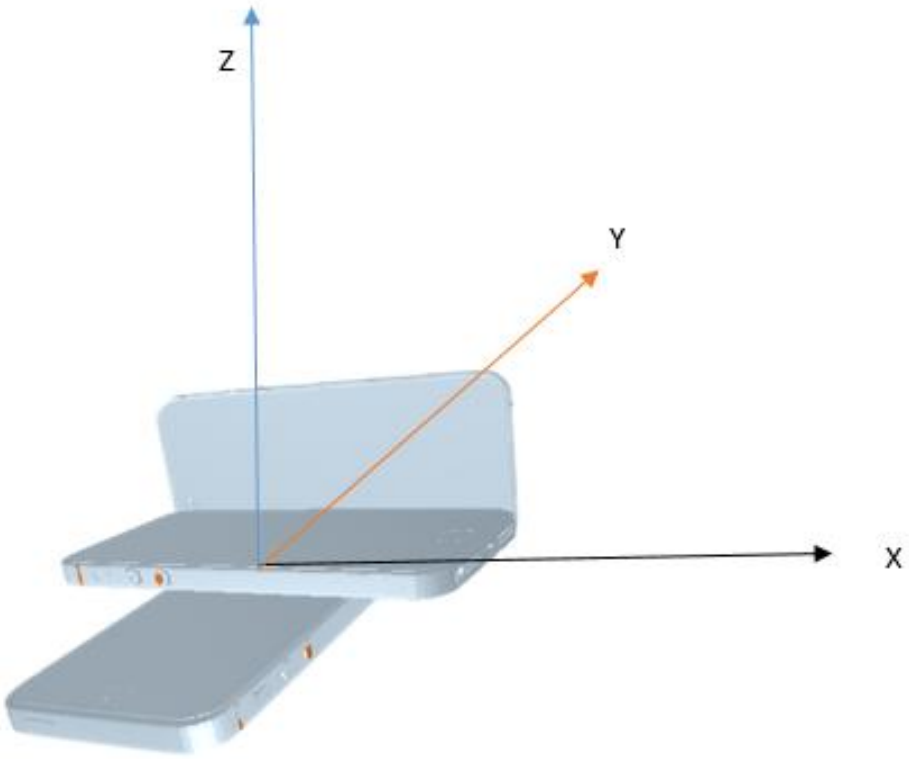


Figure 16. Orientation sensor testing

These tests were only pre-elementary and were in no way representative of actual use case. In the Spring 2016 semester, when the algorithm was ready for testing, the Android app was giving poor results while the iOS apps were faring very well. Some of the issues we were having could not have been detected in the development phase but were only discovered thanks to the technical prowess of one of the researches (stakeholder). Some issues we found and ways we fixed them are:

Bug	Reason	Fix	Person hours taken
The orientation readings from the android app had been accumulating a lot of errors.	The drift is typical of a compass reading. Which led us to believe the API we used was internally relying on it.	Issue was fixed using a different API call that called sensor type ROTATION_GAME_MATRIX. This call did not rely on the compass reading as much and was the most accurate.	20
The timestamps of the photo in the app was not exactly the same as the instant the photo was taken	The default camera app was used and as such, its API did not allow us to customize the information we got from it other than the picture and when the app was launched	A custom camera activity fragment was integrated into the app that allowed us to get shutter timing information	25
Large amounts of lag after a few minutes of recording	Thread leak was discovered and sensors were continuing to collect information in the background way after the sensors were closed	Thread handling was improved	10

Table 2. Last minute bugs

As you can see from some of the bugs listed above, it took about 55 man hours in the final stages of the project lifecycle when we are expected to finish wrapping up the project and finish testing. So having a testing strategy from the start is important. It is also a good idea to give live detailed demos to the stake holders after each scrum so the issues can be found early on. We had tried to use a variety of devices for our testing samples.

5. Summary

In conclusion, this workflow has been successful in our particular application mostly because it has been received well from all parties: stakeholders, team-members and team leads. If you belong to any of these categories in your project, you are encouraged to try the methodology if you believe implementing this could benefit your development process to be able to produce faster results, make well documented code and easy to transfer projects.