**Gardening Application**

Joseph Podnar – Scrum Master / Project Owner

Wenchao Wang – Quality Assurance Manager

Glenn Selker - Software Developer

Muneeb Alvi - Software Developer

Charles Smith - Software Developer

July 8th, 2015

**CS 1530 SPRINT 3 DELIVERABLE**

**Project Process and Progress**

The white board is one of the greatest tools in a programming team’s arsenal. It was essential in the development of our goals for sprint three. Much of the user interface and basic user experience was yet to be nailed down at the end of sprint two. The white board helped us each draw out our differing ideas and explore the merits of each. Our team had a 2.5 hour meeting at the beginning of sprint three to help develop a unified vision for the next step of our product.

There was significant disagreement about the layout of the garden bed user interface. Some members wanted to have a persistent key on a frame that would divide the screen into two vertical sections. This would allow a key to be displayed, and provide space to place the buttons on. Others argued that, as we currently have no zoom feature implemented, it would cut out too much of the viewing space. The key idea was dropped for the time being. The buttons were moved into a menu. While this would require a persistent bar at the top of the screen, it actually improved the user experience. This bar was present at the menu screen and within all species based activities. Leaving the bar up created a unified feel throughout the product.

Further disagreements arose when discussing the garden bed mode of adding, editing, or removing plants. What should be displayed and how? In this case after much discussion we decided to include the persistent two-panel design we had rejected for the general garden view. There are a lot of moving parts in the add/edit/remove mode. Expecting the user to realize the menu bar options had changed was an unreasonable belief. Our new design required less clicks and less thought.

It was this new user interface for add/edit/remove that provided one of the greatest challenges for sprint three. Any time the group has worked on a new piece of the user interface, it has taken longer than projected. This is no fault of any coder, rather it is the nature of working on a new platform. Our garden bed UI programmer ambitiously accepted ten user stories related to this feature. Upon the completion of sprint three, not one of these user stories were completed. Many will be completed early in sprint four, but the scope of our goals for sprint three was simply too large.

We had a few challenges with code integration, but much of this was alleviated with our overarching goal for sprint three. This sprint had a significant focus on peer review. In the previous sprints, our Quality Assurance Manager handled all code reviews. In this sprint we decided each member of the group should be assigned at least one pull request to review. Reading over other team member’s code allowed correction of misconceptions that would have otherwise caused faults in integration.

**Code Review Experience**

The experience in code reviewing went well for me. I was told that I needed more comments and tests, enabling me to expediently add them. The new tests and clarity allowed me to find a few bugs in the process. I also enjoyed looking at other team members' code to ensure that everyone was at the same standard in coding practices. Having new sets of eyes on my code was quite revelatory. Glenn found a significant bug in my code. I was returning a clone of an object in one method where it should have returned the original object. My code improved in functionally due to these reviews.  
- Muneeb Alvi

Learning how to code for Android has been quite a struggle for me. Creating things in a new environment can take me a long time. The best moments of intuition and expansion of my ability have come through this sprint. Looking over other group members code gave me the spark of comprehension that had been lacking in previous iterations. While I am sure that my comments on the code I reviewed were a great help in improving internal code quality, I may have gotten more from the experience than author of the code.

- Joseph Podnar

I performed code reviews for at least two pull requests this sprint, and I must say that I rather enjoyed them. When you are looking at someone else's code and not your own, you see all sorts of ways to make it better. The experience was kind of fun. I definitely felt that I had a positive impact on the code quality through code reviews.

- Glenn Selker

This sprint certainly had its ups and downs with the addition of code reviews.  Our group had everyone review at least one pull request as we felt that it would help us write our own code.  These code reviews helped us get better code into the codebase, but at a large cost of time.  Some PRs would wait up to five days before a single comment suggesting changes was posted on it, so our codebase grew much slower than normal.  Reviewing other people’s code made me look at my own code much differently. I immediately made changes to my code after thinking about how the other code needed changes.  In the end I believe the code review changes we made for this sprint helped us with our code quality more than our previous sprint method. We used to have our QA member, Wenchao, review all PRs. The improvements to code quality came at a cost of velocity.

- Charles Smith

**Completed User Stories**

As a user

I want my garden to be saved to storage I can reach

So that I can move the garden between devices

Difficulty: 4

As a user

I want to see the Specicies dialog display when I click a species from the ViewSpecies activity

So that I can make choices about my selection

Diffculty: 16

As a user

I want a Species dialog back button

So I can go back to the ViewSpecies activity without changing anything

Difficulty: 2

As a user

I want a Species dialog delete button

So that I can delete the currently selected species

Diffculty: 4

As a user

I want all plants, including those outside of the viewing area to be draw

So that circles don’t need to be drawn every time I move the screen

Difficulty: 2

**Garden Application Code**

The code for this project can be located at the following address:

https://github.com/cs1530-group1/garden\_app/tree/

**User Story Choice**

The customer had been less responsive over sprint three. Significant details such as the real world measurements of her garden prevented work on creating real representations of plants. It is for this reason that the main focus of sprint three was to focus on user interaction. The scope of the goals set for this sprint was far reaching. A total of fifteen user stories were chosen to be worked on. These upgrades to our project focused on two main themes: displaying information about a clicked species and the editable mode for plant placement\movement. While displaying species information was in the product backlog, much of the user stories describing the layout and behavior of the editable mode were not.

Displaying species information was an important goal. Our original plan for the user experience was placing plants directly from the species screen. It was and still is an essential part of creating a garden. However, after some discussion, it was decided that this would not be the only place a user could add new plants. We based this idea on the assumption that if a user planted one instance of a particular species, it would be very likely they would wish to place another. We wished to expedite this process. A user story was added for being able to place a new plant of the same species of a currently selected plant in the garden bed. This brought the total number of add/edit/delete based user stories to ten. While much work was completed for the backbone of these stories, no story from this list was completed.

One unique story that received a lot of attention had to do with the location of storing our garden information. The client wanted to be able to share information about her garden with others. Our solution was to place the garden data in a user-accessible location. With much of the code review being splint among the whole group, it seemed like an ideal task for the Quality Assurance Manager to complete in his spare time. The task was more complex than initially envisioned. It required understanding how android treats internal and external storage. We ran into trouble with emulated devices not having emulated external storage. Still, this work was a positive progression for the project.

**Quality Assurance**

# Emulator Device Specification:

Resolution: 1280x800

CPU: Intel Atom (x86\_64)

Target: Android 5.1.1 - API Level 22

RAM: 1536 MB

SD Card: 1024 MB

Our client uses a Samsung Tab 4, so the specs are as close as the actual device. As for this sprint, the app only has to run on this device with no significant defects. SD card is required to test storage functionalities of the app.

# **Defects in Garden App**

Garden App is the front end of the app, so most of the tests are done manually in Android emulator. Unit tests are run if such tests exist.

### **File IOs may fail if the permission to read/write on external storage is denied**

**Status**: Closed

**Reproducibility**: Rarely

**Priority**: Low

**Description**: Found by code review. App ops (available on many custom ROMs and rumored for future versions of stock android) revoking permissions from this app will cause it to fail every time. This defect cannot be reproduced on stock Android. It can be reproduced on emulator by not setting an SD card. This is low priority because it is not needed at the moment. YAGNI.

### **FileOperation.load does not trim the white spaces and newline characters**

**Status**: Fixed

**Reproducibility**: Always

**Priority**: High

**Description**: Found by unit tests. FileOperation.load() always returns a string with a newline feed at the end. This caused test failures when the input string does not have a newline in the end. Fixed by trimming the string before return.

### **Layout in ViewSpeciesInfoActivity cutting off information**

**Status**: Fixed

**Reproducibility**: Occasionally

**Priority**: Medium

**Description**: Found by running the app on emulator with screen size less than 5”. When the app is in View Species Info, part of the text is displayed outside the screen. Fixed by setting layout weights so the display is scaled evenly at all screen sizes.

### **Garden resets upon returning to the main activity**

**Status**: Fixed

**Reproducibility**: Always

**Priority**: High

**Description**: Found by running the app on emulator. Upon returning to main screen, the garden is reset. All changes made to the garden are discarded. Fixed by adding code to only load the garden once in main activity.

### **Minimizing the app in View Garden causes the app to crash**

**Status**: Open

**Reproducibility**: Always

**Priority**: Medium

**Description**: Found by minimizing the app when the app is in View Garden. Trying to bring the app back up causes the app to stop. Not fixed yet – cause of crash is unknown, may need some time to look into it.

### **Main activity lacks unit tests**

**Status**: Fixed

**Reproducibility**: Always

**Priority**: High

**Description**: Found by code review. There is no test for loading garden from storage in main activity. Fixed by breaking onCreate() into small methods and adding tests for these.

### **Drawing activity takes longer to start**

**Status**: Open

**Reproducibility**: Always

**Priority**: Medium

**Description**: Found by running the app on emulator. There is a slight performance issue where the app freezes for a second before the garden view shows up. The issue is currently being investigated.

# **Defects in Garden Engine**

Garden engine is tested with Unit Tests and code inspections.

### **Garden.getSpeciesInfo returns a reference of the specie in garden**

**Status**: Fixed

**Reproducibility**: Always

**Priority**: High

**Description**: Found by code review. getSpeciesInfo() should return a pointer to the copy of the species found instead of the specie inside the garden. Fixed by changing the method and creating a clone of the object.

### **Garden serialize and deserialize has issues dealing with datetime**

**Status**: Open

**Reproducibility**: Always

**Priority**: High

**Description**: Found by unit tests and code review. Garden.gardenToString() uses 24-hour date time format but Garden.stringToGarden() uses 12-hour date time format.

### **Species.java does not have getters and setters for the fields**

**Status**: Open

**Reproducibility**: Always

**Priority**: Medium

**Description**: Found by code review. Currently Species class does not have getters and setters for its fields. They should be added.

### **Garden.getSpeciesInfo throws a NullPointerException when the species to be returned has a null plant date or prune date**

**Status**: Fixed

**Reproducibility**: Always

**Priority**: High

**Description**: Found by unit tests. The method is supposed to return a copy of the species with the specified name. It throws a null pointer exception when plant date or prune date of such species is null. Fixed by modifying the code in the method to check for null conditions.