Project Plan

Unified Butterfly Recorder iOS — May1614

Matt McKillip
Mason Berhenke
Blake Burns
DJ Todd
Kyle Long
Yu Jin
Eric Soland

Advisor: Dr. Diane Rover
Client: Nathan Brockman
Team Website: http://may1614.sd.ece.iastate.edu/

Contents

1 Introduction

1.1 Background and Problem

Originating in January of 2013, the first Unified Butterfly Recorder (UBR) team set out and created a mobile application on Android that enhances data collection during butterfly surveys. By using a mobile application instead of filling out paper forms, it is possible to easily gather information from across the world and analyze it much more quickly. The third UBR team is currently working on setting up the tools necessary to unify the application's data storage in a central server, storing worldwide information in a single location. Their central server will allow organizations of all sizes to easily upload and view survey data from other teams using the app.

The problem our UBR team has been faced with is to create an iOS version of the existing UBR Android application, and to implement all of the new data sharing services that UBR3 is currently adding.

1.2 Definitions

Please note the following definitions, as they will be used throughout the rest of this document.

- Sighting - Records about a particular butterfly.
- Survey - A group of sightings and records about the environment of those sightings.
- Organization - A group of "MyUBR" users. Certain members of an organization may be able to see surveys uploaded by other members of the organization.
- Partner - A third party with whom individual users and organizations have chosen to share survey data with.

1.2.1 Surveys and Survey Protocols

Together, the existing mobile apps provide an excellent platform for gathering butterfly survey data. The mobile apps allow users to create various types of surveys. The type and method of data collection during a survey is dictated by protocols. Each protocol is designed to answer a specific type of research question. There are many protocols used today, including the following:

- Presence-Absence - Simply record whether a particular species has been sighted in an area.
- Distance Sampling - Record the distance of sightings from a specific line or point transect in order to estimate local distribution and abundance.
- Meandering Survey - This protocol involves individuals or groups walking an indeterminate path looking for as many individuals as they can find.

- Pollard Walk - A surveyor repeats an identical transect (route) different times over the course of years, recording sightings visible within a specific range of the path, aiming for consistency of recording.

The mobile apps allow users to create and save surveys according to each of those protocols.

1.3 Objective

By referencing the existing UBR application on Android, our objective is to create an iOS application that will have all the same functionality while also adhering to the iOS design principles. Along with implementing current functionality, our goal includes working with the UBR3 team to implement the new ability of uploading survey data to a central server.

2 Deliverables

In order to accomplish the objective described above, we plan to focus on one primary deliverable with a few secondary deliverables, should we reach all necessary checkpoints ahead of time.

2.1 UBR for iOS

Our primary deliverable will be an iOS app that will have all available features of the current UBR app for Android. In addition to the features of the Android app, we will also need to incorporate a new back end web service supplies by the UBR 3 team which will back up and record all surveys, as well as handle user authentication. In order to show that this has been delivered, the app must pass Apple's app screening process and be given the go ahead to be released on The App Store.

2.2 Secondary Deliverables: Android and wearable support

Should the team be able to accomplish the above deliverable ahead of time or without the aid of all team members, secondary deliverables such as updating the Android UI and implementing a wearable version of UBR may also be completed.

2.3 Android

With the current UBR app on Android already being functional, most changes made to the app will be cosmetic. Creating a more user friendly and modern interface that will help users understand the primary purposes behind the app.

2.4 Wearable

In addition the the deliverables described above, we may also create a wearable interface for hands free surveys. This will be implemented on an Android based watch because Android watches work with both Android and iPhone users and will thus hit a larger portion of the market.

3 Market Analysis

Our target audience are scientists who are interested in surveying butterflies. Many surveyors still use pen and paper to track their data. The UBR app will allow these surveyors to record their findings quicker than paper, and eliminate the need to manually input the data to a computer. By using the UBR app, the user will also be able to take advantage of its cloud features to upload their data and view data on the database.

61 percent of US citizens have smartphones, and the number keeps growing. Many surveyors will already have a smartphone available. The ability to more quickly and easily manage their surveying data will entice them to download the UBR app. A well liked Android app already exists. Many more want to adopt the product, but do not have an Android phone. By creating an iOS app, more users can have the chance to use the UBR app.

3.1 Survey Database

UBR 3 is currently developing a backend to store and utilize surveying data. By hooking the iOS app to these endpoints, it will enable surveyors to easily store and share their finding to others. This database will follow the BAMONA's design specifications to work well with their butterfly and moth database. The Android group has already determined which data fields that surveyors are interested in, and the iOS app will follow that model.

3.2 Competitors

The Google Play store and Apple App Store have surveying apps. Android apps, such as Magpi have outdated UI, and don't seem to focus on the speed of data input. The UBR apps main focus is to make data entry as quick and easy as possible. These feature will be a strong advantage against the current market.

4 System Requirements

Many scientists and citizen scientists around the world will use the UBR iOS app to survey butterflies. It is important that this iOS app is intuitive and easily understood to reduce time to train in using it. This app should also be convenient to use, so citizen scientists will be more likely to survey butterflies.

If the requirements change, this section will also. Reiman Gardens retains the rights to change the requirements as necessary.

4.1 Primary Requirements

- App must store at least 1000 sightings without access to wifi, or mobile data to send information to database.
- It should take less than 15 min for a novice to become comfortable using this app, at least for basic surveys.
- It should take less than 15 seconds once the app is open to do a survey on a single butterfly.

4.2 Secondary Requirements

- Upgrade app to include voice to text for butterfly names with the correct species getting in the top 4 options at least 95 percent of the time.
- Modify current android app to be more similar to iOS app.

4.3 Non-Functional Requirements

App should look interesting and pleasant while following iOS design standards. A survey may need to be used to confirm this abstract requirement.

4.4 Resource Requirements

App must work on at least 95 percent of iOS phones on the market, at least as of September 2015.

- Development Tools
  - XCode
  - GitHub
- Trello
- Mac computers
- Slack chat

5 Proposed Solution and Assessment

5.1 Swift Language

Swift is the latest language by Apple used to create iOS apps. Multiple avenues were looked at including some application builders such as HerokuApp, but

these were rejected due to the poor performance of such apps. While we could have used Objective-C and Apple's Cocoa framework, Swift is a new modern language that is easy to pick up and allows for powerful app development. Since it is easy to learn, we can spend less time with learning curves and more time in development. Objective C will be used when necessary however.

5.2 Realms for iOS

Realms for iOS is a mobile database library. Unlike Sqlite, Realms is object oriented, making it easy to manage and maintain in code. Realms targets Swift 2.1, but will also work with 2.0 and 1.2. Since this UBR iOS app is being started from the ground up, we can use Realms with Swift 2.1. Using Realms will further allow us to focus more on the product rather than technical hurdles.

5.3 Parse SDK

To be able to work with the UBR Server Project Plan, we will be using Parse SDK to easily access the database. Since this is an outside requirement, we don't have much say in how we will be accessing this. Thankfully the UBR3 team is optimizing this.

5.4 Android Wear SDK and WatchKit (Tentative)

If the wearable stretch goal is met, we will most likely be using the Android Wear SDK in order to create an easy user experience with Android smartwatches. Swift's version of Android Wear is WatchKit, which is easily integrated with Swift.

5.5 Open Source Swift Libraries

5.5.1 MapKit

The Map Kit framework provides an interface for embedding maps directly into your own windows and views. This framework also provides support for annotating the map, adding overlays, and performing reverse-geocoding lookups to determine placemark information for a given map coordinate.

We will use this framework to provide a visual representation of the users survey. Based on our clients feedback, this is a very useful and important deliverable.

5.6 OpenWeatherMap

OpenWeatherMap is a service that allows developers to easily add weather information to mobile applications on both Android and iOS. The weather information returned includes temperature, humidity, cloud coverage, wind speed/direction and a precipitation forecast. It is very lightweight since it won't require us to install anything on our app, we will simply use an HTML GET call based on

the user's location to request weather information from the OpenWeatherMap API.

5.7 OpenEars

The OpenEars framework provides an easy to use SDK to work with voice recognition on an iOS app. It is written in Objective C and can interface with Swift. It also adds the capability of using offline voice chat to allow surveyors without data connection to still use the service. How reliable this is when it comes to scientific names requires some research.

6 Validation and Verification Plan

6.1 Fall 2015

Our primary goals for our first semester are to research UBR1 and UBR2's applications, gather requirements, and spend time researching and learning iOS development. To achieve these goals we will be in constant communication with our clients at Reiman Gardens, our advisor, and the past UBR senior design teams. We will need to make sure we consider all parties when gathering the requirements

Our plan is to meet biweekly with our clients at Reiman Gardens to ensure our client is happy with our requirements and able to give us additional feedback for additional requirements. Due to our large team, we will be meeting with our advisor in a set of two meetings to get feedback on our progress. Finally, to meet with past teams, we will be hosting Google Hangouts when necessary.

We plan on developing prototype applications based on the requirements we have gathered so far. We will discuss the prototypes during our meetings so we can improve our requirements based on the feedback received.

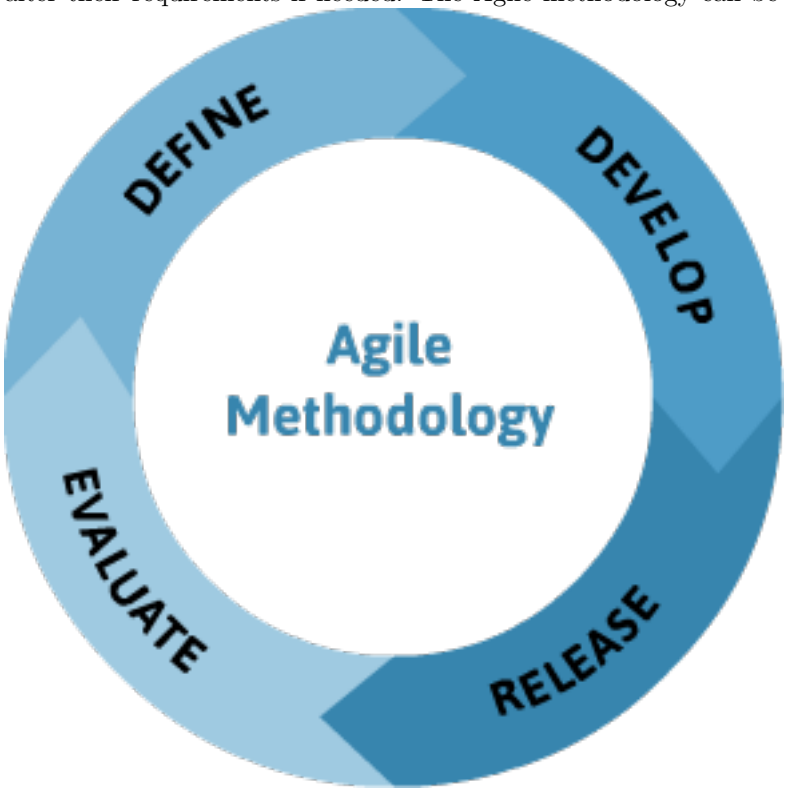
Holding frequent meetings will help us gather strong requirements for our application development.

6.2 Spring 2016

Once we have gathered our requirements, we will begin developing our final application. Our team will have two methods of validation and verification. The first will be presenting the client with our current application every two weeks. The second method will be using test-driven development principles.

We will be developing using the Agile methodology. The agile methodology requires that work be completed in a series of sprints, which our team has decided to be two weeks in duration. At the beginning of each sprint we will decide which requirements need to be implemented or updated in the code. Each requirement will be represented as a user story on our Trello board. A user story helps keep the use case, goal, and requirements in mind when developing. Trello is an online collaboration board that will hold our user stories. Once the sprint is over, a retrospective meeting will be held between our team and the

clients, to ensure that any new functionality is implemented correctly and that the system still fulfills the requirements. This process will help ensure that the client is happy with the direction of the development, and allow the client to alter their requirements if needed. The Agile methodology can be seen below:

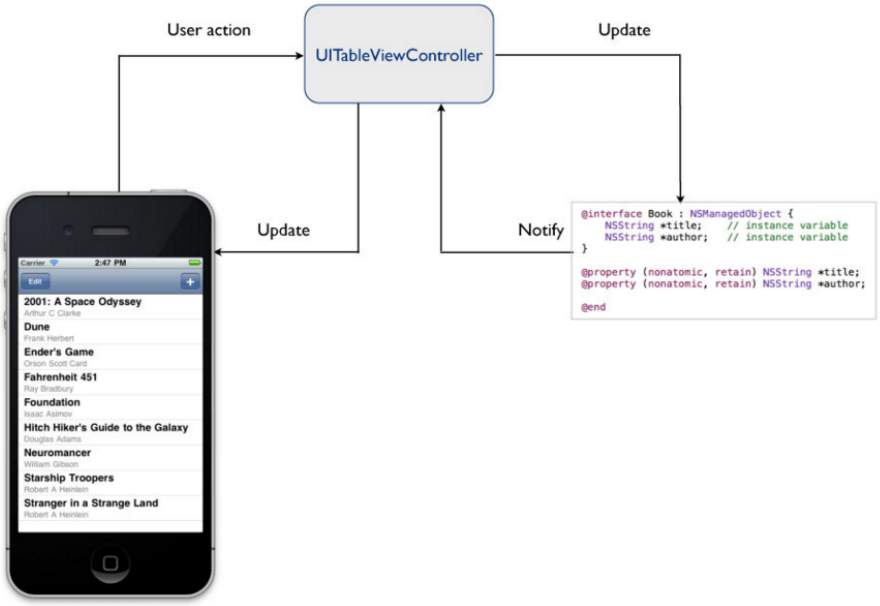


Our second method of validating our requirements will be using test-driven development. Test-driven development is a software development process where a developer writes unit test cases before writing code, then will write the code until all of the unit tests pass. Test-driven development process provides the benefit of significant reduction in bugs, built in requirement verification, and a lower development time. We have also connected our code with a Continuous Integration server. The server will build and run our unit tests every time the code is pushed to the repository as well as every night. Our team will benefit by immediately seeing if the code they pushed caused bugs in the application.

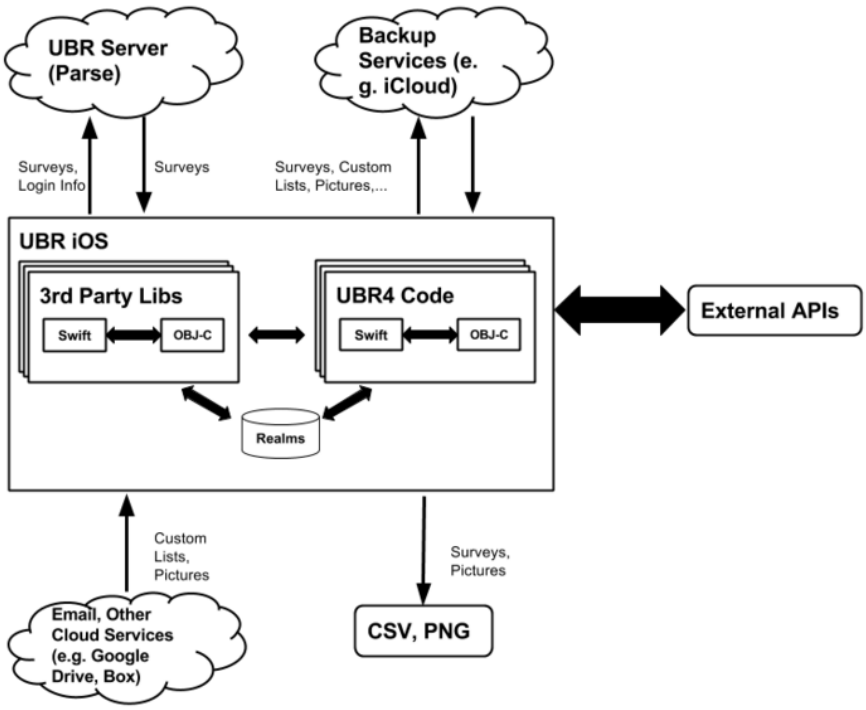
Our two methods of validating and verifying our requirements have been set up during the fall semester, and will hopefully allow us to provide our clients with a high value product, by the end of the spring semester.

7 System Structure

From an app standpoint, iOS is strictly held to an MVC design pattern: it is very difficult to stray away from this when coding for application (regardless of using Swift or Objective C). An example of how MVC works in iOS is shown below.



The entire app as a whole will consist of Swift and Objective C libraries working hand-in-hand with our own code. This app will then communicate with several cloud services that we will not have a part in coding such as the currently in progress UBR Server. There will also be ways to export the data as well as several external API calls to applications such as weather. Below is an image representation with specifications as to what type of data is sent/received by services.


UBR Server (Parse)
Backup Services (e. g. iCloud)
Surveys, Login Info
Surveys
Surveys, Custom Lists, Pictures,...
UBR iOS
3rd Party Libs
Swift
OBJ-C
UBR4 Code
Swift
OBJ-C
External APIs
Realms
Custom Lists, Pictures
Surveys, Pictures
Email, Other Cloud Services (e.g. Google Drive, Box)
CSV, PNG

8 Work Breakdown Structure

8.1 Team Structure

With this big of a team, we will be splitting into sub-teams, each of which have specific deliver ables. There will be sub-team leaders as well for each deliverable, and these may change depending on the sprint.

8.1.1 Overall Team Structure

Member	Position
Mason Berhenke	Team Leader
Blake Burns	Communications Leader
DJ Todd	Key Concepts Leader
Yu Jin	Webmaster
Matt McKillip	Core Team Developer
Eric Soland	Core Team Developer
Kyle Long	Core Team Developer

8.1.2 Sub-Teams

Core iOS Team

Member	Key Component
Blake Burns	Key Survey Data (e.g. weather)
DJ Todd	UI Flow of Application
Matt McKillip	Visual representation of survey
Eric Soland	Exporting data from the phone
Kyle Long	iOS Realms - Data Management

Innovation Team

Member	Key Component
Mason Berhenke	Voice recognition and research on iCloud usage
Yu Jin	Importing CSV lists to iOS

8.1.3 Team Experts

Member	Topic
Mason Berhenke	Innovation
Blake Burns	Swift Language
DJ Todd	User Interface
Yu Jin	Agile Methodology
Matt McKillip	Testing in iOS
Eric Soland	Repository (Github)
Kyle Long	Parse database language

8.2 Current Time line

Date/Time span	Milestone
August - October	Define Project, Deliver-ables, and Sub-teams.Learn Swift basics individually.
October 15th	Basic website done, fully functional.UI looks secondary priority
October 26th	All members have full understanding of surveys.
November 9th	Pre-alpha layout. Hello World Apps and Basic Swift Understanding Done
November 20th	Basic app (buttons, layout, flow), testing infrastructure
December 1st	Individual prototype apps completed
January 15th	Integrate individual prototype apps
January 30th	Working 'bare bones' prototype
February 29th	Application released for beta testing

9 Risk and Mitigation Strategies

Risk	Mitigation Strategies
Unable to complete wearables app.	Have two teams, one focused on innovation and one focused on core app functionality.
Unable to complete core application functionality.	Have two teams, one focused on innovation and one focused on core app functionality.
Focusing on side stories rather than the main stories	Frequent meeting with clients, advisor's and in-between team members to discuss on what should be done for the duration.
Start coding too late	Decide on what to do and set a time limit for it.
Code Reviews	Assign people to read up on code and using slack to run the program daily.
Design Architecture leading to un-maintainable code	Follow standard iOS MVC design pattern.
Feature creep	Prioritize features by importance, and assign features to team members
No unit tests	Code reviews and builds before submitting code to source control

9.1 Feasibility Analysis

All members of our team are seniors in software or computer engineering. While only 2 have any experience with iOS design, the whole team have started on coding early by creating a basic hello world app with iOS and reading up on iOS design. Thus, while it will be a learning experience for the team, everybody is committed to completing the project.

The project itself can be accomplished in the time allotted to us. For the first semester, we will focus on reading up on iOS design and learning the basics of the iOS design. Tasks are also assigned to every team member. This tasks are related to functions that we want to be used on the app. A basic implementation of each function will be completed by the end of Fall 2015 semester.

For the Spring 2016 semester, using UBR1 Android application as a reference, all our individual codes will be compiled together and added to a iOS UBR app. If time permits, further innovations will be added to the iOS app that will be extension of what the Android app have and updating the Android app to the same level of functionality of the iOS app.

10 Conclusion

The completion of this project will greatly increase the number of UBR users. Organizations have been waiting to adopt the UBR Android app until a partner iOS version comes out. Having more organizations on UBR is very exciting because it will increase the amount of data collected from surveys. When the iOS version is complete, the amount of surveys on UBR will greatly improve the research being done on butterflies in the United States, and throughout the world.

11 Current Project Status (Updated 11/17/2015)

11.1 Development

Currently each member of the team is working on an individual prototype for our application. We have set a due date of 12/1/2015 for each member to present their findings and their progress. The prototypes each member have been working on can be found in the team structure table. Once we have individually developed a feature, we will start the integration process, which we hope to have done at the end of the semester.

11.2 Team Infrastructure

Our team has also been working very hard to get our development infrastructure up and going. Our team has created a Github repository for our code. We have also started mapping our development using Trello. Additionally we have all started using Slack, which is an integrated messaging application. We have set up automated messages for whenever code is pushed to the repository, a card is moved in trello, or our code is build in Bitrise. Bitrise is a continuous integration server we have set up that builds our code on a virtual machine and runs all of the tests. We have set up for this to happen every time a commit is pushed to the dev branch and also every night. Bitrise will help us catch errors early in the development process.

12 Similar Applications

12.1 Wildlife or Animal Tracking Apps

- Bugs count
- Record Wildlife
- Wildlife Log
- Africa: Live
- Sightings Tracker
- Bird Atlas Recording Softwar
- Kruger Park Sightings

12.2 General Mapping and Geo-location Apps

- GPS Grid Reference
- GPS Waypoints Navigator
- Gmemo for Field Survey
- Gmemo Survey
- GIS4Mobile
- 2GIS
- Maverick
- Path Tracking