Final Project Report

Stephen Baier
Brandon Davis
Nathan Gross
Leah Shanker

Revision 1.1
Last revision 5/12/2011

Table of Contents

Team Hakuna Matata

Team Hakuna Matata was formed as a Northern Arizona University Computer Science senior design project team. Each member of Hakuna Matata has studied Computer Science for at least a few years and know their way around computer systems. However, it was the unique combination of the specialized skills of each member which enabled the team's success. The members of the team and a brief synopsis of their background and skills are as follows:

1. Stephen Baier

Stephen Baier has extensive experience with user interface design with a particular emphasis in games. His academic electives include Virtual Worlds, Game Production, Advanced User Interfaces and Computer Graphics. He's developed in C and C++ in UNIX, Windows and OS X environments and Swing GUI in Java. Stephen has some familiarity with embedded and mobile programming and strongest trait is his ability to write code.

2. Brandon Davis

Planning for a future in Game Production, Brandon Davis has placed an emphasis on communication and the overlaying architectures of any system. Brandon has been exposed to the architecture of various systems through courses such as Software Engineering, Game Production and Data Mining. He's familiar with a variety of languages, including C/C++, Java, PHP, and Actionscript. With a minor in English, Brandon also focuses on being able to visualize and explain the overall problem and solution. Communication and abstraction are key components to any successful software system and these are what Brandon excels at.

3. Nathan Gross

Nathan Gross has an emphasis on web development, software architecture, and game production. His elective courses include Web Development, Virtual Worlds and Computer Graphics. His language familiarity includes PHP, Javascript, Perl, ASP, Java, C/C++, and Web 2.0 tools. Nathan is ready to prove that he is versatile and can perform well in many kinds of positions.

4. Leah Shanker

Leah Shanker's particular academic interests include User-centric Software Development and Embedded Systems Programming. Her ability to quickly grasp complexities, solve problem efficiently and effectively lead numerous successful teams during her academic career prime her for leadership on this team. Her Computer Science electives taken include Advanced User Interfaces, Embedded Systems, Virtual Worlds, Artificial Intelligence and Computer Security.

Project Overview

1. Climate Change

One of the greatest questions facing modern society is whether or not life on Earth is sustainable. Sustainability is the ability to endure over time. If the environment is destroyed then life would not be able to endure. If this destruction doesn't get reversed then life on Earth would eventually cease to exist.

This concern of sustainability is largely a result of deforestation. Deforestation is the removal of a forested region or large amount of trees in order to convert the land for farming. According to the World Resources Institute, more than 80 percent of the Earth's natural forests already have been destroyed. Additionally, up to 90 percent of West Africa's coastal rain forests have disappeared since 1900. Deforestation, in turn, causes additional effects such as drought and famine. Without trees and plants soil becomes arid and dry, making it much more difficult to plant there in the future. Additionally, trees are natural consumers of carbon dioxide, so a decrease in trees increases the amount of carbon dioxide in the air. This overabundance of carbon dioxide decreases the overall air quality.

The decline in air quality is a result of increased emissions of carbon dioxide in addition to the reduction of trees, which would naturally offset these emissions. Techniques used in these countries cause the deforested land to remain unusable for future crops since the land becomes devoid of nutrients necessary for cultivation. This means the population must move on and deforest additional land in order to plant more crops. There are several companies attempting to reverse the effects of deforestation and teach improved farming techniques to these communities. One such company is our sponsor, Clean Air Action Corporation.

2. Clean Air Action Corporation (CAAC)

Clean Air Action Corporation focuses on developing and implementing low-cost solutions to improve air quality and reduce air emissions of their clients. CAAC has implemented their own cost-effective solutions for several companies across the United States to help these companies meet air quality control standards or to reduce their environmental impact. These companies include large corporations such as General Motors, Shell Oil, and Ford Motors along with many other smaller companies.

CAAC helps these companies in one of two ways. Either they find innovative methods of reducing air quality concerns such as installing combustion controls, suggesting gasoline additives that reduce emissions, or upgrading distribution and transmission techniques. If they can't provide their client with an innovative solution then they offer emission transactions to offset the harmful emissions.

An emission transaction is when the client purchases carbon, Nitrogen Oxide (NOx), or VOC credits, depending on what emissions they want to offset. These credits are representations of the absorption effect that trees have. For example, trees are able to absorb carbon dioxide; if a large amount of trees are planted then they can be used to offset carbon emissions for the client. The trees can be quantified in order to verify how much carbon form carbon dioxide that they are absorbing. A client can then purchase a verified carbon credit in order to compensate for emissions that they are producing.

These methods have been very successful for Clean Air Action Corporation's clients. The Connecticut Resources Recovery Authority even noted that because of CAAC they "saved approximately $7 million in capital expenditures, achieved full compliance with the Nitrogen Oxide RACT standard on time, and, in fact, over the past eighteen months has surpassed the regulatory requirements and prevented an additional 220 tons of NOx from being emitted into the air, thereby creating approximately 200 Emission Reduction Credits for marketing to other facilities," which shows the strength and impact of just one of CAAC's solutions.

CAAC has helped clients attain emission controls, but has also identified another problem in underdeveloped nations. CAAC has helped reverse the problem of deforestation while also helping to secure their business model. Along with The Institute of Environmental Innovation, they formed one of the world's largest and most successful reforestation and economic development programs, TIST.

3. The International Small Group & Tree Planting Program (TIST)

The International Small Group & Tree Planting Program, or TIST, focuses on empowering Small Groups of subsistence farmers in countries such as India, Nicaragua, Kenya, Tanzania, and Uganda to reverse the effects of deforestation, drought, and famine. TIST encourages farmers to form small groups that plant and care for large groves of trees. These groves are then verified and the GhGs absorbed by the grove can be calculated as carbon credits and then sold.

Once a small group of farmers has been formed, they contact TIST indicating their intent to plant some trees. TIST sends a quantifier out to inspect the area, confirms the size of the grove, confirms that the land is owned by the group, and collects other relevant data (like what types of trees will be planted). The quantifier also takes multiple GPS perimeters of the grove for verification. Information such as area, perimeter, density, and shade coverage are calculated and stored. The quantifier then returns to an Internet cafe and uploads the grove data to the TIST database server.

Once a year, a quantifier will be sent to return to the grove and re-quantify the tract. This is called a secondary audit. Auditing involves first downloading the original tract data at an Internet cafe. Then the quantifier makes their way to the grove to collect the data once again. This data is verified with the original data to ensure that the size of the grove hasn't changed and that the trees are healthy. After a grove is verified by TIST, the data collected can be used to calculate how much carbon dioxide the trees are absorbing, which is then converted to the virtual commodity of carbon credits. CAAC can then find buyers for these carbon credits.

This process provides income to both the quantifiers who are employed by TIST and the small groups that maintain the groves. Small groups receive small stipends for every tree that they continue to maintain. This encourages them to provide care and attention to ensure that the trees stay alive as well as providing income. TIST has been very successful in their endeavors to both provide income to local farmers and to help reforest areas in need. The TIST program began in only Tanzania with less than 40 small groups of 10-12 people each. Now TIST spans over 6 countries in East Africa, Asia, and Central America. TIST now includes over 60,000 participating members, and over 10 million trees have been planted as a result. On average, about 6,000 trees are being planted by TIST small groups every day.

4. TIST's Current Implementation

In order to verify the carbon credits, quantification must occur. Quantification is where a TIST quantifier visits a grove tract, an area where trees will be grown, and audits it in order to gather information about the tract. The current TIST quantification process contains several inefficiencies. The quantification process includes:

1. Receiving tract data from the TIST servers via an Internet cafe. This data is stored on a Palm device.
2. The quantifier must then leave the Internet Cafe and head out for the actual grove, which can take a while.
3. Once the quantifier reaches the grove then they must gather the perimeter of the grove in GPS points as well as additional information that confirms the maintenance of the grove. This is the actual Grove Tract Quantification.
4. After this is completed, they must then return to the Internet cafe.
5. Once back at the Internet Cafe, the quantifier will sync their Palm Device and upload data back to the TIST Server.

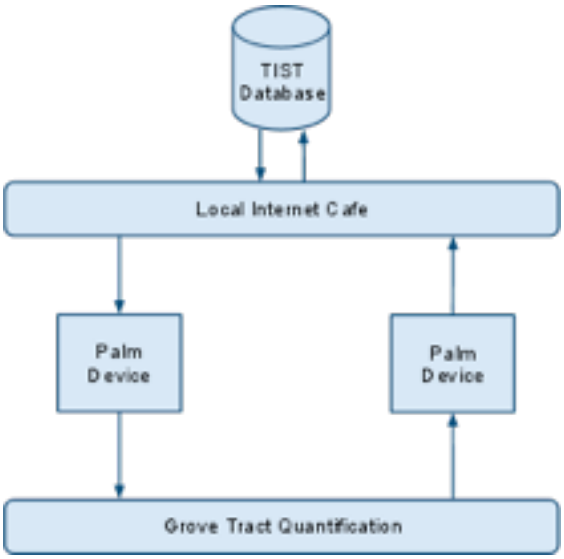

Figure 1. Quantification Process

This process is inefficient for several reasons. Initially it is apparent that requiring an Internet Cafe is a large time overhead. However, since there are little-to-no wireless Internet signals in most of these areas, they must commute to the local Internet Cafes. In addition to this, the actual gathering of data at the grove can be extremely time consuming. This is due to the current software used to collect the GPS locations, Acreage.

Acreage suffers from several faults that can lengthen the overall process considerably. These faults range from simple hindrances to causing a complete restart of the quantification process. In addition to this, the faults can't be fixed because the developers of Acreage did not distribute any source code. The major implementation issues of Acreage include:

- Data Restriction
  - 750 point limit
- Error Prone Nature
  - Fatal errors cause device crashes.
  - Poor Bluetooth support causes numerous fatal errors.
- User Interface Flaws
  - Buttons that have no function
  - No reversal of action on GPS collection
  - Awkward context switching between multiple applications
  - Navigation towards a tract is done by awkwardly manipulating GPS points
- Extensive Training Requirement
  - Training quantifiers to use the software takes multiple weeks due to the numerous amounts of errors that they may encounter as well as the above UI flaws.
  - No Source Code / Documentation

5. Improvements

The TIST program has outgrown Acreage, their current application for GPS Quantification. Improvements upon Acreage could help TIST in many different ways that would save them a large amount of time and money overall:

- **Training**
  - Training costs are currently higher than normal due to the error prone nature of Acreage. A large portion of the training manual is dedicated to errors and how to handle them when they occur. Handling these errors would significantly reduce this training cost.
  - The navigation between multiple applications isn't straightforward so requires training before use. Ensuring that the UI flow is a smooth process would also save training costs.

- ○A user interface that doesn't function as expected also increases the training time for quantifiers. User feedback and functioning UI elements would ensure that training times are reduced.

- **Quantification Time**
  - ○If an erroneous point is collected, which can happen often due to GPS error or User error, the entire tract must be recollected. Allowing undo would reduce this large time overhead in these situations.
  - ○The error prone nature also dramatically increases quantification time. These errors can cause losses of data, requiring retracing their steps again, or repeatedly cause the device to restart which means the quantifier must stand still and wait on the device. Error handling would remove these delays.

- **Future Revisions**
  - ○Currently there is no source code or way for Acreage to be improved or fixed. This means that the only way to revise the software is to completely rewrite new software. This has a huge time cost for even simple revisions that may be accomplished. However, if the new software was extensible for these revisions or fixes, it would not take nearly as long as an entire rework.
  - ○Since there is no documentation for Acreage there is no way to know exactly how they implemented their features. Significant research is necessary before understanding the system, therein increasing time it takes to revise the system. A well-documented system would help future developers extend the system much quicker.

Development Process

In order to accomplish the task of rebuilding Acreage, we played to our team's strengths to utilize our resources wisely. We created a work environment conducive to development, utilized time wisely and made decisions by prioritizing completion of the project. With the steep learning curve and unreliable nature of our project's outdated hardware and software, it was extremely important to efficiently adapt to any unforeseen issues as they arose.

1. Development Environment

GroveTrotter was to be developed for two different pieces of hardware, the Centro and the Zire71 handheld devices. In order to accomplish this we set up two different respective development environments on Windows XP PCs. One of these PCs was to be used for Centro development and one for Zire71 development.

However, we soon realized that the differences between the two hardware choices was not that difficult to overcome. The main differences were simply port numbers or protocol variables. So, with these two development environments we then split them into two different task stations. The Centro environment was used primarily as a Pendragon Forms / scripting station and the Zire71 environment was used as the PalmOS C development station. This allowed us to keep the development separate but also closely linked.

Tasks were then typically handed out based on the environment. The learning curve of understanding both environments made it a logical choice to keep people within one environment. This abstracts away the difficulty and creates "experts" even within our small team. Nathan took primary control of the scripting task and handling the Pendragon Forms end of the project. He was the primary researcher and manager of the scripting side of GroveTrotter. The PalmOS side was much larger and more difficult, so Brandon and Stephen used pair programming techniques to accomplish this implementation. This style encourages one programmer to be sitting down coding while the other one is watching guiding the code and thinking at a higher level. This helps to debug as code is written and get multiple understandings of the same problem. Leah was available to help out either station as necessary, ready to take over and help implement or debug when needed.

In order to keep these multiple environments synchronized there were a few main tools used. Bitbucket and Mercurial were used to manage the repository of code. This meant that when a PalmOS feature was implemented on the Zire71 environment, it could push this new feature to the repository and then the Centro environment could pull it off the repository, making sure that the code base stayed the same throughout development. Additionally, Dropbox was used occasionally in order to transfer our documentation and other resources from computer to computer. This enabled quick syncing and transferring of large amounts of data as necessary. Links to each of these tools are provided in the references section of this document.

2. Design Methodology

In addition to the tools and multiple environments used there was a need to adapt throughout the implementation of GroveTrotter. In order to accommodate this adaptation GroveTrotter was developed using an agile SCRUM design methodology. This methodology enforces the use of a feature backlog, features that need to be implemented and features that need to be prioritized, a weekly scrum of development, and then a weekly task report detailing that week of development. This meant that after each week the task report detailed what features were implemented, what problems arose, and if there were any changes in the development needs along the way. Then the feature backlog was updated accordingly in order to prepare for the next week of development. This evolving agile design methodology allowed GroveTrotter's development to evolve and change in order to meet the challenges that arose along the way.

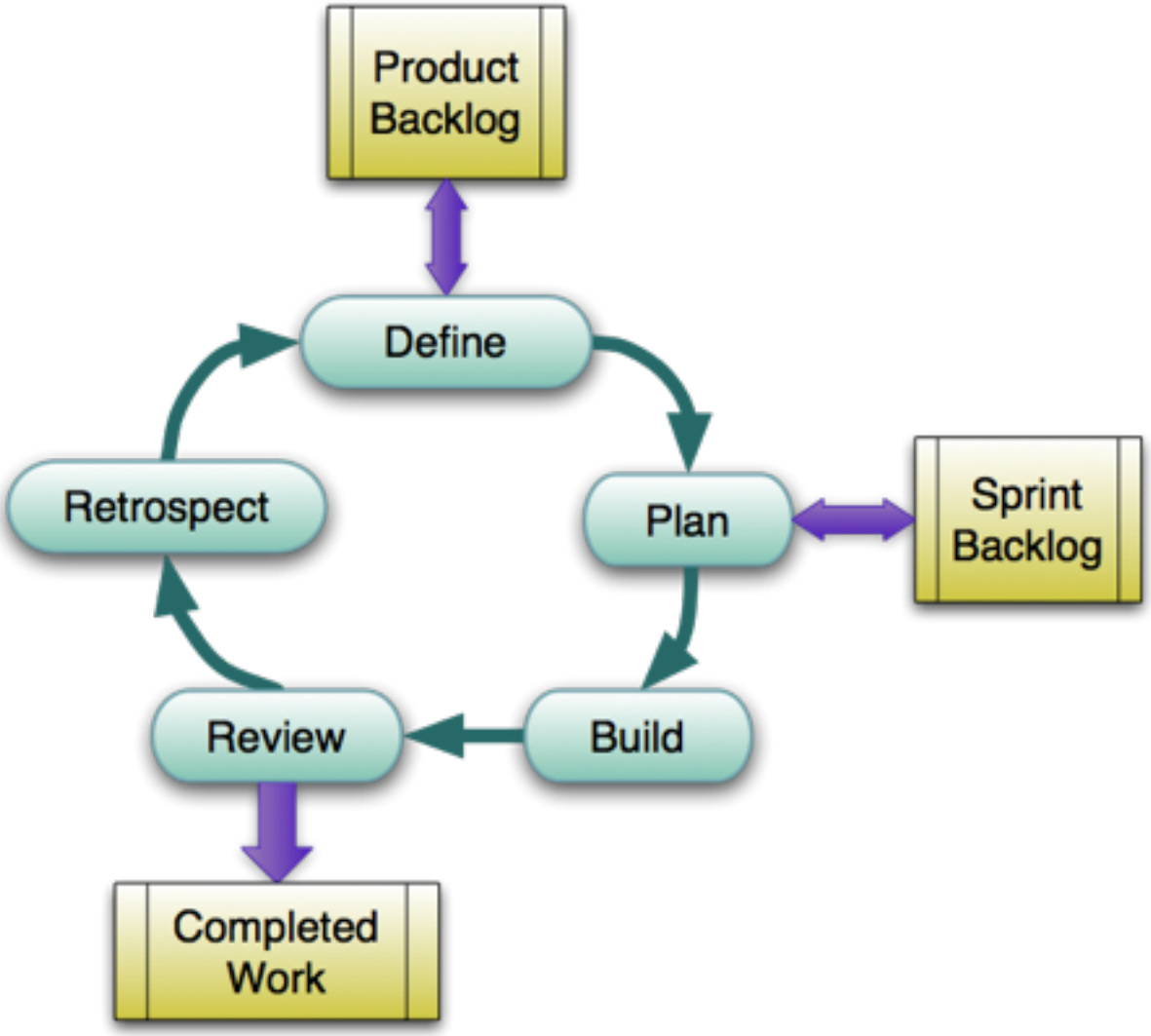

Figure 2. Agile SCRUM Software Development Cycle

3. Documentation

The design process also contributed several artifacts of documentation. These documents detailed our progress throughout the design and implementation of GroveTrotter. Each of these documents are available online through our website. A link to the website is located in the references section. The submitted documentation is as follows:

- **Requirements Document - Updated February 7, 2011**
  - Details TIST's needs
  - Functional requirements
  - Iinitial plan for completing GroveTrotter

- **Design Document - Updated May 2nd, 2011**
  - Detailed plan for building GroveTrotter
  - Includes classes, methods, architecture, and more implementation details
  - Also includes enhanced timeline of development.

- **Final Report - Updated May 12th, 2011**
  - Summarizes all information regarding the history of GroveTrotter's implementation.
  - Includes requirements, design, and implementation

- **Application Documentation - Updated May 12th, 2011**
  - Gives an in-depth examination of GroveTrotter including classes, methods, how the system works, how the classes interact, and more implementation details.
  - Also includes rationale of why implementation decisions were made, why these classes and methods exist and explaining the thought process behind the implementation.

- **Weekly Task Reports - January 25th, 2011 through May 4th, 2011**
  - Weekly reports detail previous, current, and future tasks to be completed. Most important for highlighting slippage and changes in developmental focus.

4. Timeline

Highlighted below in Figure 3 is our project timeline as a Gantt chart. Each bar column represents a month of the development cycle and each of the modules have been aligned with the amount of time it took our team to complete it.

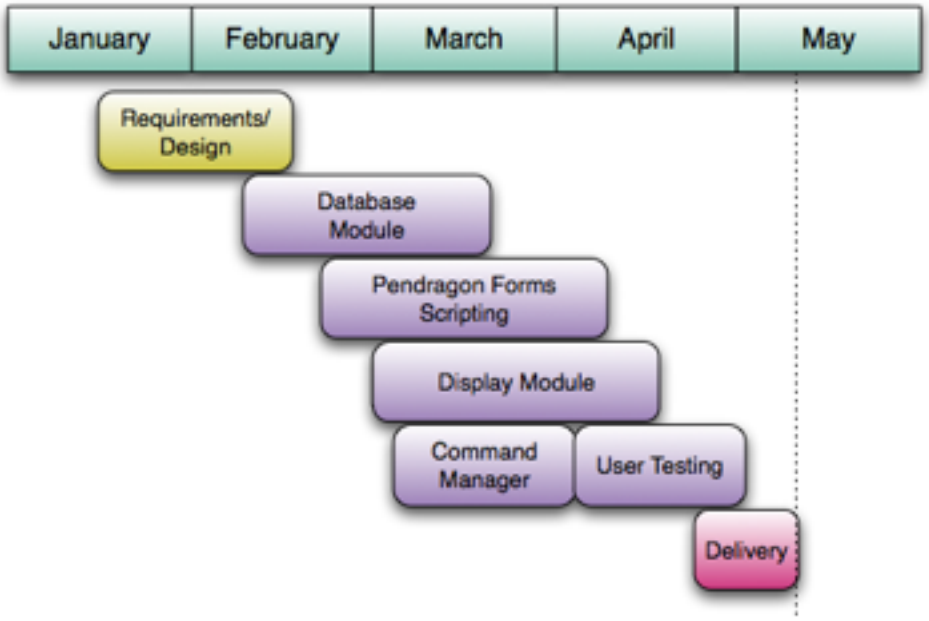

Figure 3. GroveTrotter Project Timeline as a Gantt Chart

The Requirements/Design phase was focused mostly on contacting the client, acquiring functional requirements for the project, learning the new development framework and constructing documentation. The Database Module was the first piece the team worked on, with database reading and writing utilities. We branched out into Pendragon Forms Scripting to develop the initial User Interface prototype of the project. Then, we developed the framework for the Display Module including drawing grove tracts to the screen and scaling the points correctly. At the same time, we developed the Command Manager to parse in XML commands and fire off the associated subroutine. We then got in contact with the client for User Testing and made significant changes to the project based on the results. Finally, we cleaned up the codebase and produced the documentation to prepare for final delivery to Mike Kasper.

In mid-March, we ran into significant development snags with the Display and Database Modules which set us back by about a week of development time. We responded by cutting down the time reserved exclusively for User Testing - this was made possible because Mike was able to conduct Usability Analysis after we constructed materials for him. At the very end of the development process, we began to have issues with our Magellan Serial GPS device. We handled this by just testing on the Palm Centro device instead, but a week later we began to have hardware issues with the VisionTac Bluetooth GPS device as well. Sadly, this meant we could only test our program code with simulated test values instead of actual GPS data read from the GPS device.

Requirements

We started our requirements acquisition process by communicating with our client as soon as the project started. We arranged several conference calls with our client contact, Mike Kasper, to discuss the project requirements and investigate his issues with the current software implementation, Acreage. In addition, we walked around the grass field outside the Engineering building as an attempt to become more familiar with the Acreage software and generate a list of possible improvements in GroveTrotter.

1. Overall Goals

The overall goals of the GroveTrotter project were to replace the existing functionality of Acreage and significantly improve upon it. The general functionality of Acreage included the ability to collect GPS points, draw grove tracts to the screen, edit/delete grove tracts, review both candidate grove tracts overlayed on top of one another and select the most accurate tract fit for the geography of the land. We aimed to improve upon Acreage by creating a completely integrated solution between the Custom Control and Pendragon Forms that would eliminate the application context switching while still designing for modularity. In addition, we set out to eliminate the system errors within Acreage all of which would reboot the Palm device.

2. Requirements

The functionality that GT needs to replace is important because if GT does not function at least as well as Acreage, or if GT is lacking key functionality, then it won't be used. The functionality that GT will be able to accomplish can be summarized in these requirements:

- Integrate with current TIST work-flow and existing data.
- Collect GPS points and store them in a grove tract data structure
- Draw grove tract to the screen during collection
- Allow two tracts to be reviewed at a time
- Allow selection of the best fit grove tract
- Improve error handling to reduce device failures
- Well-commented source code and plentiful documentation

3. Constraints

Constraints from the client included requiring us to develop for Palm hardware in Garnet, a deprecated software framework. Documentation was very difficult to find as most of the websites have been taken down. Additionally, we had a very limited screen size and needed to target non-English-speaking users from a broad range of cultural backgrounds. In the very end of the development cycle, we also had to switch to using simulated testing values because our provided GPS units ceased to function. This limited our ability to thoroughly test the application.

The Solution - GroveTrotter

1. Overall Solution

Our solution to the problems TIST is facing with their current software is to replace the functionality of Acreage, while addressing as many of these faults as possible. Our system, GroveTrotter (GT), must replace the existing functionality of Acreage while allowing for a more extensible system. Currently, Acreage has no source code available. GT will instead be highly documented and as modular as possible. This will allow future extensions or potential usage on different platforms as possible.

2. GroveTrotter User Interface

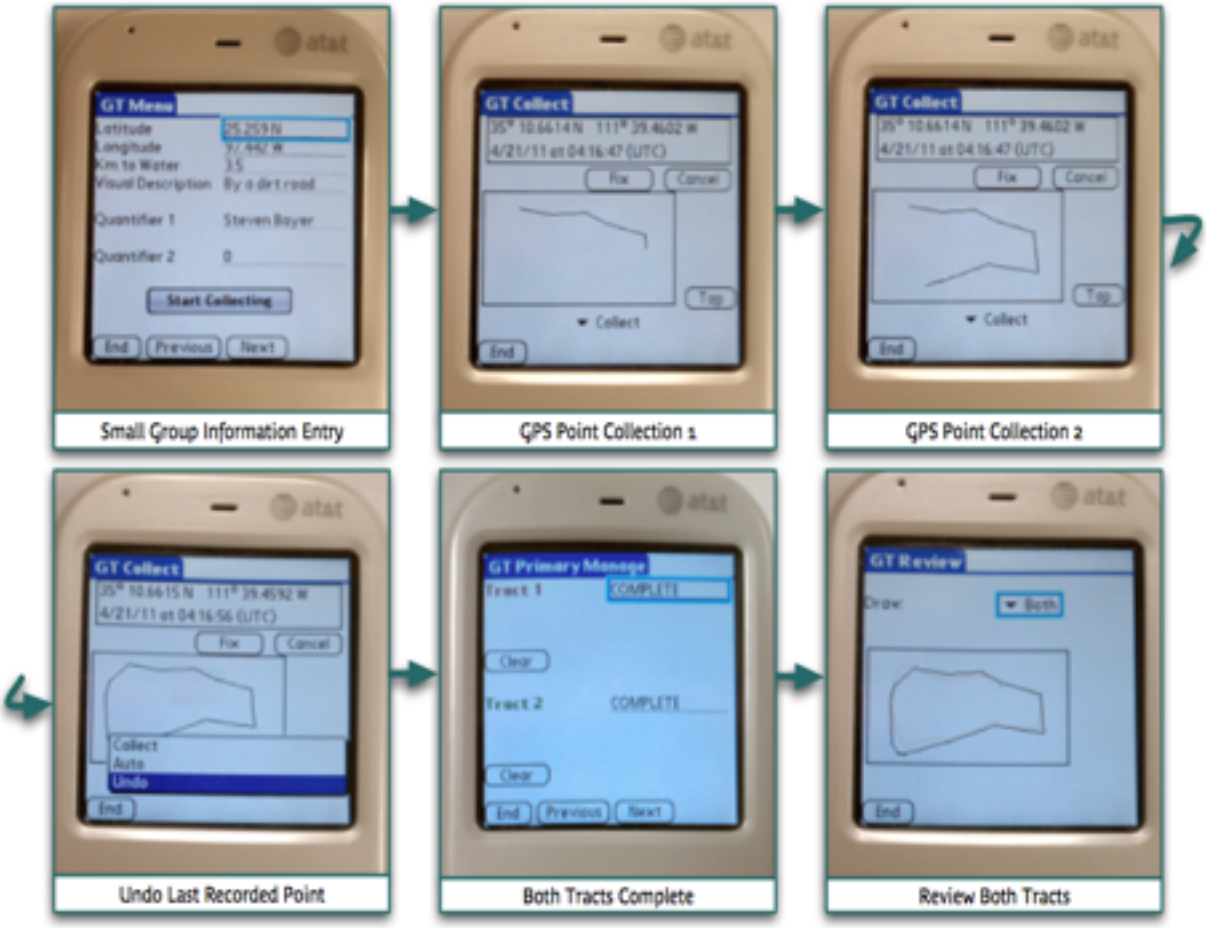

Figure 4. GroveTrotter User Interface Flow

3. Architecture Overview

Our architectural design is that of a Three Tiered architecture (3T); this is quite similar to the Model View Controller (MVC) architectural style where the Presentation Tier (top) acts as the view, the Logic Tier (middle) acts as the controller and the Data Tier (bottom) acts as the model. The difference between 3T and MVC is an explicit layering in which the presentation tier (view) can only access the data tier (model) via the logic tier (controller); see Figure 5.

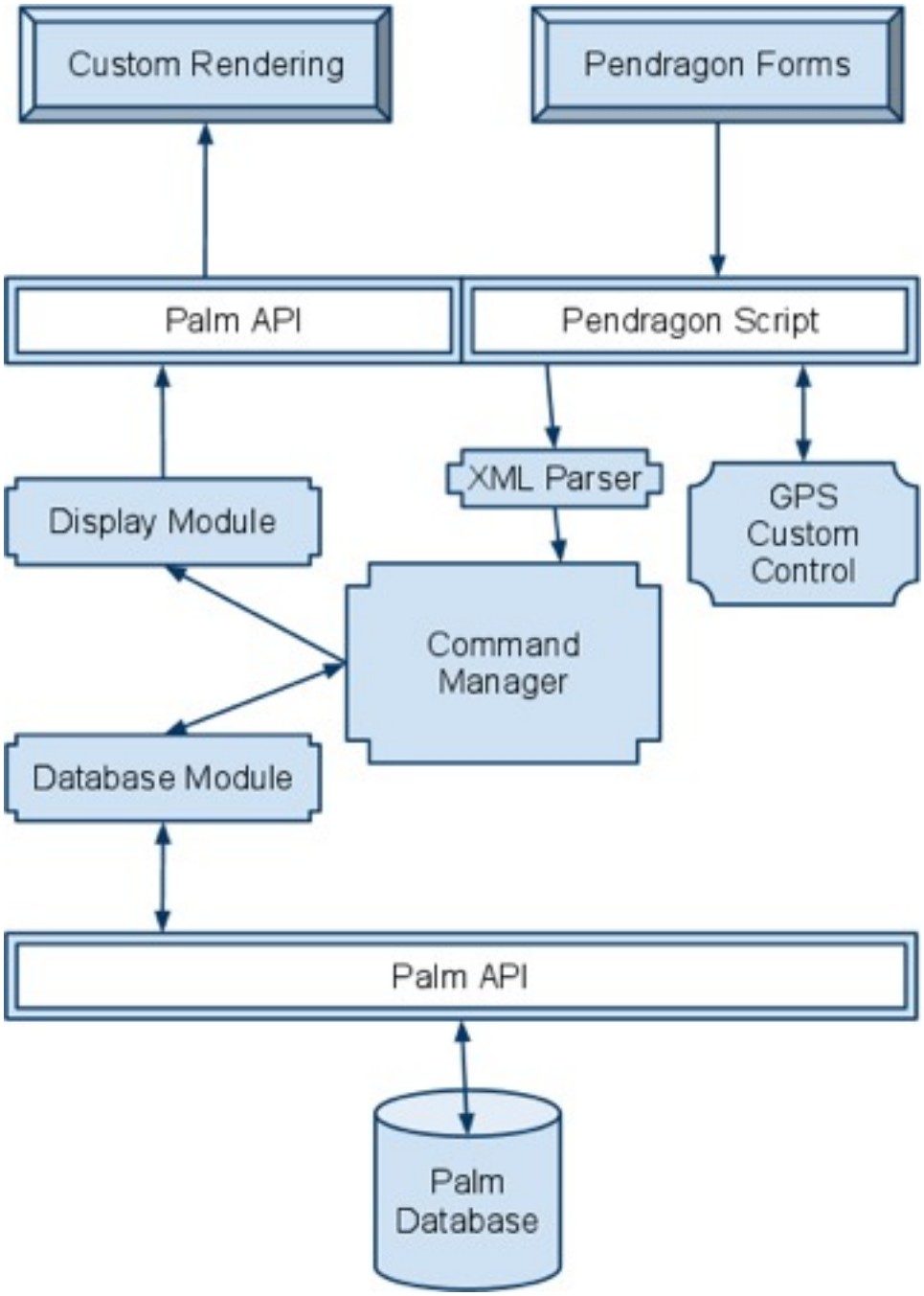

Figure 5. GroveTrotter Architecture

Presentation Tier

Pendragon Forms : This is an application on our customer's devices which provides both a user interface development environment via Microsoft Access and a communication bridge to the Logic Tier via a proprietary scripting language. The primary functions of Forms is providing user interface elements and allowing users to invoke logic via user interface controls - triggering scripts which invoke the Logic Tier. It also acts as a communication bridge between the GPS Custom Control and other Logic Tier elements.

- ●Primary user interface provider.
- ●Handles user input events.

Custom Rendering : Rendering arbitrary sets of data, such as GPS tracts, are outside the capabilities of Pendragon Forms; ergo, custom rendering is performed by GroveTrotter logic via Palm API rendering functionality.

- ●Secondary user interface provider.

Presentation-Logic Bridge

Pendragon Script : Since Forms uses a proprietary scripting language, its method of communication with the Logic Tier is simply a local memory address; anything can be placed in this memory address, but for communication purposes XML strings will be used as indicators of required action.

- ●Propagates action requests downwards (Presentation to Logic).
- ●Communication between GPS Custom Control and XML Parser.
- ●Invokes GPS Custom Control and XML Parser.
- ●Invoked by user input events via Pendragon Forms.

Palm API : The rendering capabilities of the Palm API act as a bridge between the Display Module in the Logic Tier and the user view of the Presentation Tier.

- ●Propagates rendering commands upwards (Logic to Presentation).
- ●Invoked by the Display Module.

Logic Tier

The elements within the Logic Tier are to be implemented in C and perform the necessary actions based on messages received from the Presentation Tier; these elements include GPS Custom Control, XML Parser, Command Manager, Database Module and Display Module.

GPS Custom Control : This logical element was not developed by Team Hakuna Matata nor can any additions or changes be made to it as it was developed by Pendragon Software Corp and packaged with Pendragon Forms. The GPS Custom Control handles GPS device connection of a specified port and can be queried by Pendragon Script for GPS data in XML form. The full specification of this control can be found at: http://pendragonsoftware.com/forms3/gpscontrol.html.

- ●Invoked by Pendragon Script.
- ●Populates XML string with GPS data.

XML Parser: This logical element is invoked by Pendragon Script accompanied by an XML message which indicates what action should occur.

- ●Invoked by Pendragon Script.
- ●Discovers required action based on XML tags.

Command Manager : Invokes sub-routines within the Database Manager and Display Manager based on XML Parser invocation and supplemental XML tags.

- ●Invoked by the XML Parser.
- ●Invokes the Display Module and Database Module.

Database Module : Acts as the gate keeper to the Palm Database and an abstraction of ugly Palm API calls. Performs actions such as adding data points, removing data points, querying data for rendering operations, etc; no other logical component should be able to access the Local Database without going through the Database Manager.

- ●Invoked by the Command Manager.
- ●Accesses the Palm Database(s) in the Data Tier via Palm API calls.
- ●Mutates the Palm Database(s) in the Data Tier via Palm API calls.

Display Module : Performs calculations such as converting GPS data to screen coordinates and performs rendering operations such as clearing the screen and rendering a tract.

- ●Invoked by the Command Manager.
- ●Performs conversion calculations.
- ●Performs rendering operations via Palm API calls.

Logic-Data Bridge

Palm API : To access and mutate Palm Databases, Palm API calls must be used.

- ●Invoked by the Database Module.
- ●Grants access and mutation to Palm Database(s) in the Data Tier.

Data Tier

Palm Database : Best practice in Palm development is the use of database files for storing large quantities of data. This data is stored locally as a Palm Database file (.pdb). The contents of local databases include GPS tract points along with necessary rendering data.

- ●Accessed and mutated by the Database Manager via Palm API calls.
- ●Stores local GPS tracts and rendering data.
- ●Best-in-practice data storage technique.

4. As-Built Design

GroveTrotter is an event-based application, performing no action unless triggered by user input. When a user interacts with Pendragon Forms, clicking on a control for example, the control invokes underlying Pendragon Script. This script, if necessary, queries the GPS Custom Control for current location information. The script then invokes the GroveTrotter accessory, which contains all the C logic for performing necessary tasks; this invocation is accompanied by an XML message which indicates what action needs to occur, such as point collection, tract deletion, etc. See Figure 6 below.

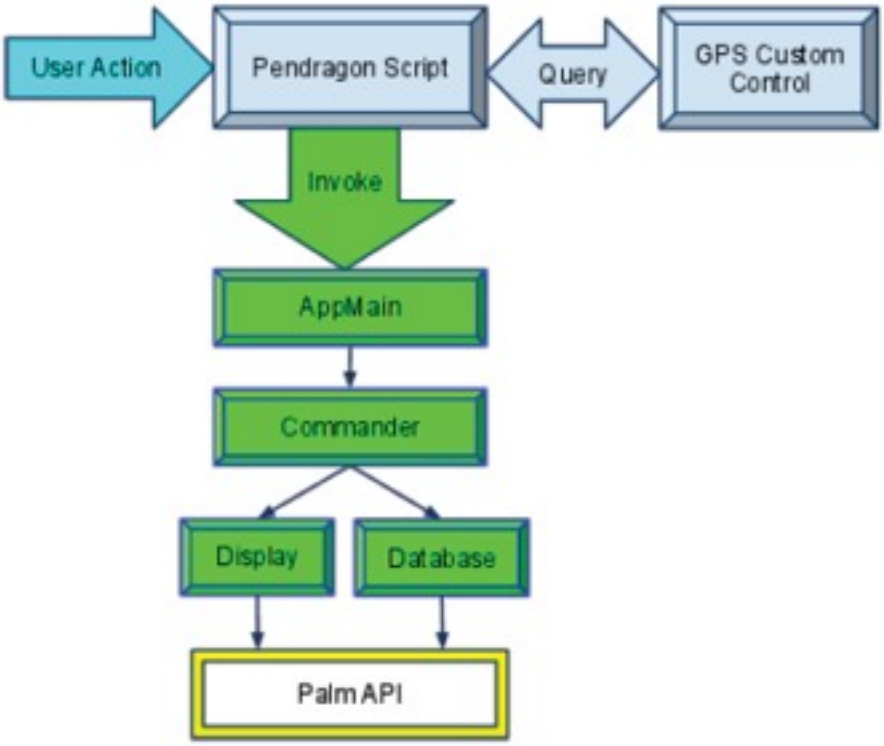

Figure 6. Application Program Flow

AppMain (AppMain.c) is always the function invoked by Pendragon Script. This function analyzes the first XML tag of the message sent to it and invokes the proper subroutine within the Commander (commander.c). Commander then further analyzes the XML message, looking for supplemental tags which indicate specifics of the action; such as which tract to add the point to and what the GPS data is. Commander then invokes subroutines within the Database Module (database.c) and/or Display Module (display.c).

The Display Module (display.c) performs conversions from Long/Lat to screen-relative coordinates. To optimize some of the calculations necessary for this conversion, the Display Module stores a rendering configuration (struct: Config) which contains information such as minimums and maximums of Longitude and Latitude, scaling values and a rectangle defining the rendering area. If a new GPS point is added that doesn't fall within the bounds of the rendering configuration, then it is re-calibrated to incorporate the new point. After conversion is complete, the Palm API is invoked to perform tract rendering. Important functions invoked by Commander:

- undo_view
  - Invoked when users are undoing point collection.
- collect_view
  - Invoked when users are collection points in either Collect or Auto mode.
- compare_view
  - Invoked when users are comparing tracts.

The Database Module (database.c) is responsible for accessing and mutating tract databases stored locally on the Palm device. It does this through use of the Palm API and Palm Databases (.pdb). Important functions:

- openDB
  - Opens a database based on the name; creates the database if it did not exist.
- closeDB
  - Closes an open database based on the database reference (DmOpenRef).
- deleteDB
  - Deletes (permanently) a database based on the name.
- createDB
  - Creates a new database with the given name.
- add
  - Adds a GPS point (struct: GPS) to an open database.
- retrieve
  - Retrieves a GPS point from an open database at given index.

Pendragon Forms v5.0 represents the front end and Graphical User Interface (GUI) of the GroveTrotter Palm Application. It consists of five main forms that are used sequentially during quantification of a grove: GT Menu, GT Primary, GT Secondary, GT Collect, GT Tract Points (this form just stores data about the the current tract and is not part of the GUI).

<u>GT Menu</u>

- Upon entering the Pendragon Forms application, users will be brought into this start up form showing meta-information about the grove location and some minor description about the surrounding area. Also, the quantifier names are stated here.
- The user flow and description fields in this form are identical to the ones used for acreage and this was done to reduce quantifier retraining costs.
- When the “Start Collecting” button is tapped, a conditional will be checked from a field on the “GT Tract Points” form to see if their is already a primary tract. If their is **not** a primary tract they will transition into the “GT Primary” form, else they will transition into the “GT Secondary Form”.

<u>GT Primary</u>

- This form handles most of the quantification work flow for collecting both candidates of a baseline tract. At the end of this form, users will be required to choose the best looking candidate.
- Using the Create button takes users to the “GT Collect” form for candidate tract collection.
- The Edit button can be used to return the “GT Collect” form for the same tract to continue collection later.

- ●The Dump button can be used to export the data to CSV which is then uploaded into an image field in the “GT Tract Points” form.
- ●The Clear button can be used to delete the tract to restart collection.
- ●After both candidate baseline tracts are collected users move on to a the review portion to compare tracts, allowing them to select the best fit.

GT Secondary

- ●This form handles most of the quantification work flow for verification and collecting an audit tract.
- ●Using the Create button takes users to the “GT Collect” form for secondary (audit) tract collection. .
- ●The Edit button can be used to return the “GT Collect” form for the same tract to continue collection later.
- ●The Dump button can be used to export the data to CSV which is then uploaded into an image field in the “GT Tract Points” form.
- ●The Clear button can be used to delete the tract to restart collection.
- ●After the audit tract is collected, users move on to the review portion to visually compare the audit tract with the primary tract.

GT Collect

- ●This form is used during the actual collection of either candidate baseline tracts or an audit tract.
- ●Users can change modes between Collect, Undo and Auto.
- ●Collect mode represents manual collection and requires the user to hit the tap button every time for a tract point to be collected.
- ●Undo mode allows the user to delete the last collected tract point just but hitting the tap button.
- ●Auto mode represents automatic collection which means a tract point is collected about every five seconds with out any user control.
- ●The Acquire button of the GPS Custom Control must be used to obtain GPS signal before tract collection can begin.

Usability Testing

1. Testing Regime

Two different usability testing methods were employed by our team and our client in order to ensure compatibility with the target users:

1. **Expert Reviews**

Although we made every effort to become familiar with the problem domain and designed GroveTrotter to match our user's needs as much as possible, it would have been impossible to become experts in the problem domain given our backgrounds and the project time constraints. Additionally, even though several of our team members have been formally trained in User Interface Design and Usability Analysis techniques, the very fact that we built the system from the ground up biased our ability to see the application from an outside perspective. In order to ensure compatibility with our target audience, we sought out the advice of experts in both User Interface Design principles and the problem domain in the form of an Expert Review, which has two parts:

Golden Rules of Usability Analysis: Our User Interface Design experts included Dr. Eck Doerry, professor of the Advanced User Interfaces course offered here at Northern Arizona University, and Mike Kasper, Northern Arizona University Computer Science graduate who was academically trained in Usability Analysis. Both of these experts provided us with a Golden Rules of Usability Analysis, which analyzed our GroveTrotter prototype in terms of the seven Golden Rules of Usability:

i. **Consistency:** Familiarity in interface "look and feel", syntax and metaphor (mental model).
ii. **Design for Evolving Expertise:** First time user tutorials, learn-ability, expert user options.
iii. **Feedback:** Every action has an appropriate reaction (indicate status and progress).
iv. **Design for Error:** Disallow error wherever possible (invalid options greyed out) or provide error detection.
v. **Reversal of Action:** Allow user to "undo" anything; avoid any permanent change.
vi. **Sense of User Control:** Users should initiate actions, not respond to them. No hidden behavior.
vii. **No Rote Memorization:** Reduce short-term memory load on users; users should not have to recall information from three screens ago.

These Golden Rules of Usability analyses gave us significant insight into the effectiveness of GroveTrotter's User Interface from a "best-practice" perspective. We then incorporated this feedback into the current prototype and future development of GroveTrotter in order to release a more polished User Interface.

Cognitive Walkthrough: We called upon Mike Kasper as an expert in the specific problem domain because he has trained TIST GPS Quantifiers out in the field during his employment at Clean Air Action Corporation. As an expert in the problem domain of GPS Quantification, he was able to provide in-depth analysis of GroveTrotter in terms of how well the application matched with the mental model of our target users. In order to conduct a Cognitive Walkthrough Analysis, our expert role-played as a GPS Quantifier using a theoretical quantification system and wrote assumptions about how the system should ideally work and what the system should be capable of without first reviewing the GroveTrotter prototype. After providing these assumptions, our expert reviewed the GroveTrotter prototype and wrote a detailed account of any discrepancies between what was assumed in the role-playing exercise and what was indicated by the GroveTrotter prototype. Each of these discrepancies were carefully reviewed by our development team and changes were either made to the application or earmarked for future development.

2. **Usability Lab Testing**

Expert Reviews were not enough to accurately gauge whether GroveTrotter would be a successful tool that could be easily understood by our target users, so we helped our client organize an effort to provide feedback from actual GPS Quantifiers in three different countries about the usability of GroveTrotter by creating a User Testing Lab Manual with a set of specific tasks that must be completed within the system. As the users attempted to step through the process of completing each task, a lab proctor took notes on any difficulties encountered by the users and wrote down any questions the users asked during the testing process. Our client selected the top three tech-savvy, English-speaking GPS Quantifiers from Kenya, India and Uganda. He trained each of these top Quantifiers to proctor the Usability Lab sessions in each respective country and assigned each proctor to conduct testing and receive feedback from 2-3 other quantifiers of lesser technical ability and English-speaking skills. The most important user feedback generally comes from the users of lesser skill who tend to have the most trouble with the application as they tend to make mistakes and ask the most questions. These mistakes and questions were precisely the kind of feedback we needed in order to identify the differences between the way the GroveTrotter prototype presents the problem and the way the target users think about the problem. Once we had a list of these differences, we brainstormed ways to better align these two models and implemented a significant amount of changes to the GroveTrotter prototype.

2. Testing Outcomes

Our Usability Testing Regime taught our team a considerable amount about the assumptions and expectations of our users. In addition, we learned a substantial amount about the specific expectations of our client in addition to identifying the functional aspects of the application with the highest priority of importance.

One of the most interesting insights we learned from testing was that our users assumed a button named “Fix” placed near the center of the grove tract display in the GT Collect form meant the button you’d press to record the center of the grove tract. In reality, it was the GPS

Custom Control annoyingly drawing this button to a fixed point on the screen. The GPS Custom Control was not directly modifiable by our team because it was included with Pendragon Forms.

The numerous application bugs and device errors were our biggest shortcoming in the GroveTrotter prototype. Both of our lab sessions reported serious issues when attempting to dump tracts, even though our team was unable to reproduce the reported errors on our testing devices. Additionally, our team had initially placed very low priority on implementing the auto-collection of GPS points feature because of the GroveTrotter prototype's already existing manual GPS Point collection functionality and the sheer complexity due to the lack of a built-in looping mechanism within Pendragon Forms. Feedback from both the target users and the client made it clear that automatic point collection was a core functional requirement, so we implemented it for the final release.

3. Exposed Issues

Our Usability Testing Regime uncovered a number of issues with the GroveTrotter prototype. Each of these issues were identified and either incorporated into the final release of GroveTrotter or tabled for future development.

1.Issues Fixed in Final Release

- Automatic GPS point collection: Created a mechanism for looping through button click events to automatically hit the "tap" button every time interval in the GT Collect form.
- Tracts should be displayed on the screen with distinctly different colors: Each grove tract is now drawn in the color that matches the color of its text in the menu. (The lines of Track 1 match the color of "Track 1" text in the list)
- Bugs with the map rescaling: Fixed map rescaling algorithm to use our floating point arithmetic - simple bug fix.
- Dump Error: Invalid Opcode field 15 or 20: Fixed a minor error in the r/w flag for the export_database() function of the Database Module - set this flag to just write. Not positive this error is fixed because we weren't able to conduct accurate testing due to hardware failure. Plus, we were never able to actually reproduce this error on our Centro or Zire71 devices even though two different Lab Proctors mentioned failures in the dump functionality.
- Allow the user to adjust the scaling of the map: Created "zoom in" and "zoom out" buttons for manual control of the map scaling of the grove tract display.
- "Acquire" and "Tap" buttons moved to the same space: These buttons are now all located down the right-hand column of the GT Collect form.
- GPS point numbers don't reset to 1 after switching tracts or deleting points: Fixed simple bug of incorrectly ordering the create & delete operations in the createGroveTract() function of the Command Module

2.Outstanding Issues Tabled for Future Development

- No "reversal" of undo/delete last GPS point: Expert Reviews suggested we shouldn't permanently delete the point when a user hits the "undo" button - we should allow the

user to reverse the “undo last collected point” action just in case the user removes too many points.

- Removal of the inane “Fix” button from the GT Collect form: This “Fix” button comes directly from the GPS Custom Control and we couldn’t locate a preference to disable the drawing of this button. One suggested method for fixing this bug is to modify the GPS Custom Control to not draw this button anymore (which would require distributing the newly modified GPS Custom Control with the GroveTrotter package).
- Switch to checkboxes/button group instead of dropdown for mode switching: Because of the very small screen real estate, we were forced to make a tradeoff decision between conserving precious screen real estate or reducing the number of stylus taps performed by the user by one per Tract collection or review session.
- No grove area calculations on the device: Lat/Long to UTM conversions are arithmetic-heavy calculations. Because arithmetic operations are very risky on the Palm devices (highly prone to error), the time required to implement and debug this portion was just outside the scope of this project. We have implemented libraries for accurate floating-point addition, subtraction and multiplication in the custom_math.c file, but a few other operations (division comes to mind) would need to be implemented first in order to begin the Lat/Long to UTM conversions necessary to calculate area on the device.

4. Usability Summary

Overall, the Usability Lab Test notes we were given from each Quantifier about the GroveTrotter prototype indicated that the application presented the problem in a manner that matched fairly well with the mental models and expectations of our target audience. Generally speaking, GroveTrotter was successful in mimicking the familiar workflow of Acreage and improved upon the design of the system considerably.

Users were particularly grateful to not have to return to the application menu and manually navigate between two different applications to complete their GPS Quantification tasks. However, a large number of system bugs were reported by the users of the GroveTrotter prototype and we’ve been diligent about either fixing these reported bugs immediately or specifically marking them down for future development.

Additionally, our Expert Review feedback was extraordinarily helpful in forcing us to rethink a number of our original design decisions and guiding us to the functional implementation changes necessary to upgrade the GUI elements in the prototype in order to develop a top-notch User Interface for the GroveTrotter release.

Future Work

There are two primary targets for future work on GroveTrotter. There is functionality that had to omitted from our GroveTrotter implementation due to the restrictive schedule. These additions could be implemented as an extension of the current GroveTrotter, which would just be a newer version of GroveTrotter. However, ideally, GroveTrotter would be moved onto an entirely new hardware that could extend the capabilities and reliability of GroveTrotter significantly.

1. GroveTrotter 1.2

The omission of some desired functionalities and improvements would be the main focus for a future iteration of GroveTrotter. The reason that these features had to be omitted from our system is also included as many of these extensions are features that were originally intended for GroveTrotter. Additionally, the benefit of each extension is included.

- Latitude, Longitude coordinate system conversion to UTM.
  - *Cost:* Much more complicated than initially expected and as a result would have taken too much time to implement for minimal benefit.
  - *Benefit:* UTM coordinates make it much simpler to perform calculations on the GPS locations.
- Area and Perimeter calculation and display
  - *Cost:* Calculating area and perimeter with latitude and longitude is much more difficult than it would be to use UTM coordinates.
  - *Benefit:* Displaying area and perimeter allows the quantifier to get a feel for how large the grove is at any point in time during quantification. Additionally, groves have to be a certain size to qualify, so quantifiers could check this in the field.
- Localization of languages and non-standard measurements
  - *Cost:* With the limited UI functionality that is within the PalmOS system, it would have been difficult to manipulate options in this way. Additionally, would have added significant overhead to release multiple builds.
  - *Benefit:* Easier and more natural use for the end-users that may not be as accustomed to English or standard measurement systems.
- Reversal of the “Undo Last Point” action
  - *Cost:* Because the “Undo” action actually removes the point from the tract database of GPS points, there isn’t really an easy method to implement this change without writing a buffer cache to sit between the tract database and the UI elements. Adding a buffer cache would add a risky element of potentially permanent data loss to an already error-prone operating system and hardware device.
  - *Benefit:* If a quantifier accidentally removes too many points from the tract, adding this feature would prevent quantifiers from having to requantify portions of the grove tract (potentially having to travel back out to the grove tract location). Additionally, it would provide better adherence to the “Reversal of Action” Golden Rule of Usability.
- Remove the inane “Fix” button from the GT Collect form
  - *Cost:* Because the “Fix” button is drawn to the screen by the GPS Custom Control to a specific set of coordinates, removing this button would require either distributing a modified version of the GPS Custom Control (thereby losing the

benefit of periodic software updates from Pendragon Forms) or writing an entirely new Custom Control.
    - ○*Benefit:* Would drastically reduce the discrepancies between the user's mental model and the presentation of the problem by the application. It would eliminate the misconception about the "Fix" button referring to the center point of the grove tract.

2. GroveTrotter 2.0

The ideal extension to GroveTrotter would be utilizing modern hardware. Hardware such as iOS on iPhone or the Android devices would give GroveTrotter many more capabilities than it currently has. Listed are numerous features that could be implemented on a stronger hardware and the benefits that they could offer.

- ●Google Satellite Overlay
    - ○*Benefit*: Visualization of a grove tract could become more than just a set of lines, it could become an actual set of lines on top of a google satellite image of the grove.
- ●Built-in GPS device
    - ○*Benefit:* Modern hardware typically includes a built-in GPS device which would eliminate the need to externally communicate with a GPS device, simplifying use of the program as well as development and time overhead.
- ●Wireless Syncing capabilities
    - ○*Benefit:* Many modern hardware are now capable of producing their own wireless signal or otherwise connecting to the Internet. If this type of hardware was used, it could entirely remove the need of the Internet cafe and speed the process overall tremendously.

Conclusion

Overall, GroveTrotter was a successful solution to TIST's problem. Utilizing the PalmOS and Pendragon Forms software was a difficult task to complete and as a result some features did have to be removed from the end product. We initially hoped to implement additional features because we underestimated the limitations of PalmOS. There are several problems and inconsistencies within the hardware itself that severely limits implementation.

According to feedback from our sponsor, they also believe that it is unfortunate that we could not implement all of the features that were initially desired. However, TIST does understand the limitations of the hardware that they use and the critical functionality is present. It should be a usable product and an improvement over Acreage regardless of the minimal omitted functionality.

Overall, the most important limiting factor of this project was the deprecated PalmOS. We as a team wish that we had better hardware to implement GroveTrotter on so that we could vastly improve upon its functionality. Hopefully TIST will be able to move to stronger hardware in the future. It would require a complete restocking of hardware in addition to a complete overhaul of their database, which would be a hefty amount of work. However, in the end, the functionality and usability increase of iOS or Android would be substantial. The biggest disappointment in the implementation of GroveTrotter was the lack of modularity to be ported away from Palm or Pendragon Forms. This means that much of the current implementation would not be usable on new hardware. Hopefully the documentation and details that we provide will be a help in future development endeavours.

References

1. Bitbucket Repository: http://www.bitbucket.org
2. CAAC clients - http://www.cleanairaction.com/clients.htm
3. Clean Air Action Corporation - http://www.cleanairaction.com
4. Deforestation statistics: http://www.nationalgeographic.com/eye/deforestation/effect.html
5. Dropbox Data Synchronization: http://www.dropbox.com/
6. Global Warming - http://en.wikipedia.org/wiki/Global_Warming
7. Hakuna Matata Team Website https://sites.google.com/site/nauhakunamatata
8. Mercurial Source Control Management: http://mercurial.selenic.com/
9. Palm Centro - http://en.wikipedia.org/wiki/Palm_Centro
10. Palm (Garnet) Operating System - http://en.wikipedia.org/wiki/Palm_OS
11. Palm Zire 71 - http://en.wikipedia.org/wiki/Zire_Handheld
12. Pendragon Forms Software - http://www.pendragonsoftware.com/forms3/index.html
13. T.I.S.T - http://www.tist.org