FieldJournal: A Mobile Data Collection and Mapping Application for PocketPCs

by

Janet W Lai

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degrees of Bachelor of Science in Electrical [Computer] Science and Engineering and Master of Engineering in Electrical Engineering and Computer Science at the Massachusetts Institute of Technology

August 14, 2004

Copyright 2004 Janet W Lai. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and distribute publicly paper and electronic copies of this thesis and to grant others the right to do so.

Author

Department of Electrical Engineering and Computer Science

August 14, 2004

Certified by:

Dr. Ceasar MacDowell
Thesis Supervisor

Accepted by:

Arthur C. Smith
Chairman, Department Committee on Graduate Theses
FieldJournal: A Mobile Data Collection and Mapping Application for PocketPCs

by

Janet W Lai

Submitted to the
Department of Electrical Engineering and Computer Science

August 14, 2004

In Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Computer [Electrical] Science and Engineering and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

FieldJournal is a mobile data collection and mapping application designed for the PocketPC brand of PDAs. It was built as an investigation into combining the data collection process and the mapping process of community planners into one piece of software, while striving to solve the specific problems of workers at the North End Outreach Network. FieldJournal has four main sections which allow the user to view data about households or individuals or to map the data. The implementation of FieldJournal follows the Model-View-Controller paradigm, separating the user interface components from the backend details. Because FieldJournal is built on the Java 2 Mobile Edition platform, many trade-offs were made in order to increase the speed of the program while still maintaining its usability. In the end, FieldJournal had only limited success in combining the two processes, as many of the more complicated mapping functions could not be included.

Thesis Supervisor: Dr. Ceasar McDowell
Title: Director, MIT Center for Reflective Community Practice
TABLE OF CONTENTS

ABSTRACT .................................................................................................................................... 2
TABLE OF CONTENTS ................................................................................................................ 3
1. INTRODUCTION .................................................................................................................... 5
   1.1 Purpose .......................................................................................................................... 5
   1.2 Problems ....................................................................................................................... 5
      1.2.1 Larger Problem – Community Planners ............................................................... 5
      1.2.2 Specific Problem – North End Outreach Network ................................................ 7
   1.3 Technical Background .................................................................................................... 10
   1.4 Thesis and General Approach ...................................................................................... 12
   1.5 Criteria for study’s success ......................................................................................... 12
2. DESIGN .................................................................................................................................... 13
   2.1 User Analysis ................................................................................................................. 13
   2.2 Task Analysis ................................................................................................................ 14
   2.3 Materials ...................................................................................................................... 18
   2.4 Procedure ...................................................................................................................... 19
   2.5 Backend Design ........................................................................................................... 20
   2.6 Interface Design .......................................................................................................... 22
      2.6.1 Introduction Section ............................................................................................... 24
      2.6.3 Household Section ................................................................................................. 27
      2.6.3 Individual Section .................................................................................................. 27
      2.6.4 Map Section .......................................................................................................... 30
3. IMPLEMENTATION (Results) .............................................................................................. 33
   3.2 View .................................................................................................................................. 35
   3.3 Controller ....................................................................................................................... 36
   3.4 Signature Capture .......................................................................................................... 36
4. DESIGN CHOICES (Discussion) ............................................................................................ 37
   4.1 Use of J2ME, CDC, and Personal Profile ....................................................................... 37
   4.2 Use of Model-View-Controller Architecture ................................................................ 41
   4.3 Zaval Light-weight Toolkit ............................................................................................ 42
   4.4 Signature Capture .......................................................................................................... 42
   4.5 Global Variables Class .................................................................................................. 43
   4.6 New Households or Individuals ................................................................................... 44
   4.7 Saving to a XML file ...................................................................................................... 45
   4.8 Notebook Model ............................................................................................................ 46
   4.9 Linked Fields ................................................................................................................ 48
   4.10 Mapping Features ........................................................................................................ 51
   4.11 Feature Attribute Update ............................................................................................. 51
   4.12 Testing .......................................................................................................................... 52
   4.13 Further Work- Accompanying Desktop Program ...................................................... 53
   4.14 Evaluating the Study’s Success ................................................................................... 54
5. CONCLUSION ......................................................................................................................... 54
6. ACKNOWLEDGEMENTS ....................................................................................................... 57
7. REFERENCES ........................................................................................................................ 57
9. APPENDIX .................................................................................................................. 59
   APPENDIX A. Table of Figures ...................................................................................... 60
   APPENDIX B. FieldJournal’s Release of Information ...................................................... 61
   APPENDIX C. FieldJournal’s Referral Form .................................................................. 62
   APPENDIX D. FieldJournal User’s Guide ..................................................................... 63
   APPENDIX E. Technology Plan for NEON: Integrating FieldJournal into the Data
   Collection and Mapping System .................................................................................. 70
1. INTRODUCTION

1.1 Purpose

FieldJournal is mobile data collection and mapping application for Pocket PC brand of Personal Digital Assistants (PDAs). It was designed to solve the specific problems of the North End Outreach Network while investigating a larger issue of how to integrate mapping into the data collection process for community planners.

1.2 Problems

1.2.1 Larger Problem – Community Planners

In the office of a community planning agency in Boston, the tables are filled with stacks of surveys used to collect information about the neighborhood, while the walls are dotted with large hand-drawn maps of the city. These surveys and maps symbolize the importance of both data collection and mapping to the work of community planners. Without the data collection process, community planners would not be able to understand the neighborhood and its residents or create maps. Without maps, community planners would not be able to advocate for the neighborhood or form crucial partnerships. Because these processes are so important, community planners are always looking for ways to make data collecting and mapping processes more efficient or to even combine them.

Currently, the data collection process starts as the community planners walk through the neighborhood, going door-to-door surveying the residents and collecting data about the
area. That data is collected on paper forms. Later, someone has to go through each form and carefully enter the data into a computerized database. Thus, data collection can be a very time-consuming and tedious job.

The mapping process begins when the community planners sort through the data and extract the relevant information to make maps. After the relevant information is put together, the community planners spread out large pieces of paper on the floor and meticulously construct each by hand. Creating these maps can take hours and even days of careful work! Later, if any information changes, the community planners have to start over and create each map again.

Much software has been developed for both processes. On the data collection side, PDA, laptop, and desktop applications aid fieldworkers in putting together large amounts of information. However, most of this software was designed for scientists or engineers and is purely data-oriented. None of the data-collection software provides any mapping functionality that the community planners need.

On the mapping side, GIS (Geographical Information Systems) is being further customized for community planners. With GIS, community planners no longer have to labor over the creation of handmade maps but, instead, can focus on the issues encapsulated by these maps. Various GIS mapping applications allow community planners to create maps and navigate through them with various panning and zooming functions. Sometimes, these maps will display some non-spatial information, e.g.
building names or street names. However, these maps cannot display a lot of non-spatial information as their primary purpose is the graphical representation of spatial data.

Based on the special nature of the data collected by community planners—large amounts of data with specific spatial components—FieldJournal is an investigation in bridging this gap between data collection and mapping. FieldJournal attempts to seamlessly integrate mapping into a data collection software application.

1.2.2 Specific Problem – North End Outreach Network

Although the general purpose of FieldJournal is to bridge the gap between data collection and mapping software, it is specifically designed for the outreach workers of the North End Outreach Network (NEON).

NEON is a collaborative effort by four agencies in Springfield, MA to better the North End neighborhood by reaching every resident and involving them in the life of the community (Waitt Family Foundation, 2003. para.2). The North End is divided up into ten zones; NEON assigns each of its seven outreach workers one or more zones. Every week the outreach workers visit the families in their zones, connecting the families to needed services and inviting them to be more involved in various community building activities (Waitt Family Foundation, 2003. para.2). In addition to this, the outreach workers collect data about each family member. The data collected includes first and last names, relation to head of household, birthdates, gender, ethnicity, language spoken, education, job status, insurance coverage, and healthcare needs. All the data is collected
on paper forms and then entered into a centralized database back at the NEON headquarters. The first time an outreach worker visits a family, he or she fills out the new information about the family. This is at least three pages of data for each family member. Each subsequent time the outreach worker visits a family, he or she notes any changes that have occurred in the original information collected. He or she also fills out a form summarizing what services he or she provided for the family. This summary form then becomes part of the data about the family.

This amounts to a large quantity of data! All the data collected serves as a file about that family which the outreach worker must have readily available whenever he or she is visiting the family. Thus, the outreach workers need to carry around the data about all the families in their zones whenever they are making visits. Carrying around the data becomes more of a burden as most of the outreach workers walk, not drive, from family to family in their neighborhood. Transferring the data collection process from paper forms to a PDA application would greatly help the outreach workers. Instead of hauling around heavy bags of paper forms, the outreach workers could just carry around a small PDA that contains all the data for his or her zone.

Currently, after field data is collected, someone has to enter each paper form into the database by hand. This is very tedious and inefficient. By collecting data on a PDA, the outreach workers could simply connect the PDA with the computer and transfer the data to the centralized database. This would greatly speed up the data collection process.
In order to create maps from this data, the GIS expert at NEON has to sort through the data and connect the spatial components to a map object. In the data that NEON collects, the spatial component is address information (street address, city, state, and zip code). To map the data, the GIS expert must connect each address to an object (e.g. building, street, or zone) on an existing map. This process has to be repeated each time a different map is made. For example, if the GIS expert wanted to make a map showing household size in the North End neighborhood, she would have to look at the address of each household in the data and find the corresponding building on an already existing map of the North End. Then she would have to copy the size value to the spatial dataset. She would have to repeat this for every household in Springfield! Later, if she wants to make a map about the number of males in each zone, she would have to do the entire process again!

FieldJournal addresses this problem by connecting each household to a map object at the time the household information is created and storing this connection as part of the data. Thus, all the data created in FieldJournal is permanently connected to a map object.

Based on the specific problems that NEON outreach workers face, FieldJournal was developed to (1) expedite the data collection process by moving that process onto a PDA and to (2) expedite the mapping process by integrating GIS into the data collection process. It is not a complete mapping program, but it addresses the problems raised when trying to map data that is collected by community planners in the field.
1.3 Technical Background

As described above, many applications perform data collection and other applications perform mapping, but none have seamlessly integrated the two processes for the needs of community planners on a PDA.

On the data collection side, programs run from a simple spreadsheet displayed on a PDA to a complicated data collection program. For example, PLog is an application sold by RockWare designed specifically for surveyors (RockWare, Inc., 2003.). PLog is basically a set of forms built on top of a relational database for a PDA. PDA Medic is another application designed for Emergency Medical Services (EMS) personnel (Pen Computer Solutions, 2004). It allows EMS personnel to collect data about the patients they treat. TracerPro is an application that allows it users to create simple custom forms, such as delivery forms, shipping forms, inspection forms, etc. (Portable Technology Solutions, LLC, 2004. para. 1). Data collection on a PDA is not a new thing. Scientist, medical personnel, delivery people, inspectors, and forest workers are just a few of the professionals that use some sort of PDA application to collect data.

On the mapping side, many programs can display maps on a PDA. The complexity ranges from a simple road map viewer to complex spatial modeling tools that are geared towards engineering and surveying professions. Usually the simple mapping programs do not have enough functionality to help the community planners, while the complicated ones are too difficult to use.
In the middle ground, somewhere between very simple and very complicated, are a host of mobile mapping applications. These applications support the basic mapping functions, such as viewing a map, navigating through the map through pan and zoom functions, creating maps from Shapefiles or other map definition files, and querying data to view and edit spatial attributes. Most of the mobile GIS applications are built on the API of the various PDA brands and can only run on that specific PDA. For example, ArcPad is ESRI’s mobile software which runs only on PocketPCs (ESRI, 2004). Similarly, PocketGIS (Pocket Systems Ltd., 2000. para. 1) and HGIS (StarPal, 2004) also only run on PocketPCs. GeoGIS (Geo Insight, 2003) and GPSPilot (GPSPilot, 2004) are mobile mapping applications that only run on PalmPilots. MapInfo has the only Java-based application, MapXTend, a mobile mapping tool which runs on all PDAs that support a Java Virtual Machine (MapInfo, 2004). However, MapXTend requires an Internet connection because it uses a server to do the map processing (MapInfo, 2004). The other programs do the map processing onboard the PDA.

Of all those mapping applications, only ArcPad can handle data collection. However, ArcPad cannot handle large datasets as its purpose is geared toward mapping and not data collection. FieldJournal, built for a PocketPC, will be the first PDA application that will handle data collection and perform all basic mapping functions.
1.4 Thesis and General Approach

FieldJournal is a piece of software that resulted from a study investigating the best way to combine data collection and mapping on a PDA. The specific problems of the NEON outreach workers are the context for this investigation.

FieldJournal was created using the Java 2 Mobile Edition (J2ME), Personal Profile and Connected Device Configuration (CDC) platform (Sun Microsystems, 2004). FieldJournal runs on all PDAs with a Java Virtual Machine that support the J2ME/CDC/Personal Profile platform. Currently, PocketPCs are the only PDA with support for that Java Virtual Machine.

1.5 Criteria for study’s success

Usability is the most important criterion in this study. Community planners must be able to use the data collection and mapping software on a PDA, otherwise it is ineffectual. Specifically, this study should produce a product that the NEON outreach workers will use. This criterion will be tested by how well FieldJournal is accepted by the NEON outreach workers and whether or not they use it.
In addition to usability, the three problems that NEON Outreach workers face (described in Section 1.2.2) must be addressed:

1. FieldJournal needs to run on a PDA.
2. FieldJournal should be able to connect to a desktop computer and synchronize its data with a database.
3. FieldJournal must aid in creating maps, i.e. the creation of spatial databases for desktop mapping.

2. DESIGN

2.1 User Analysis

The users of FieldJournal are the outreach workers of NEON. They are all native Spanish-speakers and can read and write Spanish proficiently. Their English speaking, reading, or writing is not as strong. However, they have been trained to collect data in English and are accustomed to filling out the paper forms in English. The users have little experience in using advanced technology. Most of them use a personal desktop computer but only for word processing, email, and Internet access in the graphical Windows operating system. They know how use a keyboard and type. Few of them have ever used a PDA, and none of them know how Graffitti® (the PDA handwriting protocol).

The above user analysis requires that FieldJournal follow some specific parameters. Because the outreach workers have limited English skills, FieldJournal must be similar to
the paper forms to which the workers are already accustomed. Secondly, because the workers are used to using a Windows computer, the PDA should be based on a Windows system. Thirdly, a keyboard or normal handwriting recognition must be provided. Finally, the backend should be as hidden from the user as possible. They should interact solely with the user interface front end, which should look like the paper forms.

2.2 Task Analysis

Data Collection - FieldJournal must be able to handle the following data collection tasks:

1. New Household Information

   The user needs to be able to enter information about a new household. This includes information about the head of household, address of home, and household income. Subtasks include filling out the form and setting the household location on a map.

2. New Individual Information

   The user needs to be able to enter information about a new individual. This includes information about the individual’s household, age, ethnicity, language, gender, employment, education, and health. Subtasks include filling out the form and connecting the individual to an already existing household.

3. View Existing Household Information

   The user needs to be able to view an already existing household and make changes to that information.
4. View Existing Individual Information

The user needs to be able to view an already existing individual and makes changes to that information.

5. Add new Summary

Each time he or she visits an individual, the user needs to add a new summary form to that individual’s information. The summary form must include information about the date, the individual, what the individual needed, what service the outreach worker provided, and any other notes.

6. Add new Referral form

In order to refer an individual to a specific agency, the user needs to be able to create a referral form. This form includes information about the individual, the referring agency, the date and time of an appointment, and the reason for a referral. More importantly, the individual’s signature, showing his or her consent to this referral, must be obtained for this form through a digitized signature capture feature. Subtasks include filling out the form, obtaining the signatures, and later printing out paper versions the form.

7. Add new Release of Information form

The user needs to allow an individual to sign a Release of Information form. This form includes information about the individual and the agency that is receiving the data about the individual. More importantly, the individual’s signature,
showing his or her consent to the release of information, must be obtained for this form through a digitized signature capture feature. Subtasks include filling out the form, obtaining the signature, and later printing out paper versions the form.

8. View existing Summaries, Referral, and Release of Information forms

The user needs to be able to view existing summaries, referral, and release of information forms. However, after the referral and release of information forms have already been signed, the user should not be allowed to change them.

9. Deleting a Household

The user needs to be able to delete a household and all the users associated with that household.

10. Deleting an Individual

The user needs to be able to delete an individual and all the summaries, referrals, and release of information forms associated with that individual.

11. Synchronizing the data to a database

The user should be able to connect the PDA to a desktop computer through a synchronizing mechanism (usually a USB connection). The data from the PDA should be copied to the desktop computer and put into the database. The database information should be updated accordingly, but the user should be warned each
time some data is overwritten. Subtasks include physically connecting the PDA to the computer.

12. Extracting data from the database

The user should be able to extract data from the database to put on the PDA. He or she should be able to choose which individuals or households to place on the PDA. He or she should also be able to put all the data associated with one zone on the PDA. The data on the PDA and database should be already synchronized. Subtasks include physically connecting the PDA to the computer and synchronizing the data.

Mapping- FieldJournal must be able to handle the following mapping tasks:

1. View Map- zoom, pan, and reset

The user should be able to view a map of the North End of the Springfield, MA. This should include the 10 zones, all the streets, and all the buildings. The user should be able to zoom in and out, pan, and reset the map to the full extent.

2. View Household/Individual Information

The user should be able to select certain map objects (buildings) and view household and individual data connected to that feature. Subtasks including selecting the map object, viewing a list of the households connected to that object, choosing a household to view, and choosing an individual to view within that household.
MapNotes - FieldJournal must be able to handle the following MapNotes tasks:

1. *Attach a MapNote about a map object*

   The user should be able to attach a note to a map object. Subtasks include selecting the map object.

2. *View already existing MapNotes about map objects*

   The user should be able to view an already existing note about a map object and makes changes to it. Subtasks include selecting a map object, viewing a list of notes associated with that map object, and choosing a note to view.

2.3 *Materials*

FieldJournal was implemented using the Java 2 Micro Edition’s Connected Device Configuration and the Personal Profile (Sun Microsystems, 2004). GeoTools 1.0, an open-source mapping toolkit based on Java 1, was also used to perform most of the mapping functions (GeoTools, 2004). I adapted GeoTools so that it would run on J2ME. For the user interface, I used Zaval’s open-source Light-weight Toolkit (Zaval Light-Weight Visual Component Library, 2004). An open-source PNG encoder by J. David Eisenberg was used to save the signature images to PNG images (Eisenberg, 2003). The open-source J2ME XML-parser kXML 1.2 was used to save and load data (KXML Project, 2003). The programming was done on IBM’s Websphere Studio Device Developer version 5.6 (IBM, 2004). The PocketPC’s also used IBM’s Websphere Micro Environment J9 Java Virtual Machine (IBM, 2004). All programming was done on a Hewlett-Packard (HP) Pavilion zt1175 laptop with 768Mb of memory running Windows
XP. Testing was done on a HP iPaq Pocket PC h4011 and a Toshiba Pocket PC e800. The HP had 64MB of RAM (55MB main memory) and a 400 MHz Intel XScale processor. The Toshiba had 128MB of SDRAM (32MB main memory) and an Intel 400 MHz PXA263 processor. The screen resolution for both Pocket PCs was 240 x 320 pixels.

2.4 Procedure

1. General problem formulation and backend design
   a. I decided that we want to integrate data collection and mapping. I also outlined the general design concepts.

2. Material Selection
   a. I created three prototypes using three different sets of material.
      i. J2ME with CLDC 1.1 and MIDP 2.0. This would have run on all PDAs, not just PocketPCs. For mapping, I adapted GeoTools to run on CLDC 1.1 and MIDP 2.0. For data collection, I used MIDP 2.0.
      ii. ArcPad and VBScripts. This would only run on PocketPCs. I used the mapping provided with ArcPad and wrote VBScripts for data collection.
      iii. J2ME with CDC and Personal Profile. This only runs on PocketPCs. For mapping, I adapted GeoTools to run on CDC and Personal Profile. For data collection, I used Personal Profile and Zaval’s Light-weight Toolkit.
3. Specific problem formulation for NEON and Interface design
   a. I decided to build FieldJournal in the context of the NEON outreach workers. Based on the material selection, I came up with a more specific interface design.

4. Design Testing
   a. I created a screen shot prototype of FieldJournal and presented it to the workers at NEON. With their help, I was able to further solidify the design.

5. Implementation-Testing (3x)
   a. I implemented FieldJournal and presented it to the workers at NEON. They tried using it for a few weeks, and based on their comments, I made changes to FieldJournal. I did this three times.

2.5 Backend Design

There are four main components in the backend:

1. The DBF file, which contains the spatial data for creating the map.
2. The Map, which is created from the DBF file.
3. The Table, which contains the non-spatial data collected by the user.
4. The MapNotes, which the user can create and connect to various Map objects.
The DBF file, table, and MapNotes all connect to different Map objects. (A Map object is one specific feature on the map. For example, in Figure 1, which shows first three NEON zones, each zone is a Map object.) Each row of the DBF file corresponds to exactly one Map object, a one-to-one connection. Similarly, the connection from each MapNote object to the Map is one-to-one; each MapNote object can only connect to one Map object. However, the connection from the data table to the Map is many-to-one. Multiple rows in the data table can connect to the same zone. Figure 1 shows how the first two rows connect to the first zone while the last two rows connect to the second zone.

<table>
<thead>
<tr>
<th>Shape</th>
<th>BldgID</th>
<th>Zone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polygon</td>
<td>101</td>
<td>1</td>
</tr>
<tr>
<td>Polygon</td>
<td>102</td>
<td>2</td>
</tr>
<tr>
<td>Polygon</td>
<td>103</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BldgID</th>
<th>Head of Household</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Eddie Kim</td>
<td>70 Main St.</td>
</tr>
<tr>
<td>101</td>
<td>John Smith</td>
<td>71 Main St.</td>
</tr>
<tr>
<td>102</td>
<td>Jane Doe</td>
<td>72 Main St.</td>
</tr>
<tr>
<td>102</td>
<td>Cheryl Park</td>
<td>73 Main St.</td>
</tr>
</tbody>
</table>

Figure 1. A symbolic representation of what FieldJournal does. The four main components are the DBF file, the Map, the Table, and the MapNote objects. The DBF file is the file containing the spatial data for the Map; it essentially creates the Map. The Map shows the first three zones as specified by the DBF file. The Table contains the non-spatial data collected by the user. Each row of data in this Table is also connected to a feature in the Map. Finally, a MapNote is also connected to a Map object.
The data is collected from a form interface and is stored as a Table in the backend. Thus, the Table only contains information requested by the forms. If the user wants to record information not requested by the form front end, he or she can create a MapNote and connect it to a Map object.

The user never actually sees or interacts directly with the DBF file or the Table. The user only sees the Map created from the DBF file, the form front end connected to the Table, and the MapNotes that he or she creates.

2.6 Interface Design

FieldJournal is modeled after a fieldworker's notebook. It has tabs that allow the user to navigate from one section to another. There are four main sections, as shown by the top row of tabs (see Figure 3):

1. Introduction (I)
2. Household Information (H)
3. Individual Information (I)
4. Map (M)
The Introduction section allows the user to create new Individuals, Households, or MapNotes, or to search for existing ones. The Household Information section allows the user to view the data about Households. The Individual Information section allows the user to view the data about Individuals. The Map section displays a map of the North End neighborhood and allows the user to find Households, Individuals, or MapNotes at certain locations.
Only the necessary and available sections are enabled (see Figure 3). The Introduction and Map sections are always enabled, but the others are not. In order to get to the Household or Individual sections, the user must navigate through the Introduction or Map sections. Figure 2 shows the navigation paths to the different sections.

### 2.6.1 Introduction Section

![Figure 3. The Introduction section. There are four sub-pages in the Introduction section, but only two (Intro and Search) are always enabled. The first page, “Intro”, allows the user to create a new individual, a new household (HH), or search for an existing Individual, Household or MapNote. Pressing the “search” button will bring up the third page, “Search”. The second page appears when the user presses the “new individual” button. The fourth page appears after the user has used the search functions in the third page. Note that in the top row of tabs, only the Introduction (1) and Map (M) sections are always enabled.](image)

As shown by the second row to tabs in Figure 3, the Introduction section contains four pages: (1) Intro, (2) HH, (3) Search, (4) Results. Only the Intro and Search pages are always enabled. From the Intro page, the user can choose whether to create a new Individual, create a new Household (HH), or search for an existing Individual, Household, or MapNote. If the user decides to create a new Individual, the second page (HH) is enabled. The HH page shows a list of existing Households. The user must select
a Household to which the new Individual will belong. After the user does that, the first Individual section page is then displayed (see Figure 6). If the user decides to create a new Household, the first Household section page is then displayed (see Figure 5). If the user decides to search for an existing Individual, the third page, the search page, is shown (see Figure 4). The user can also just directly click on the search tab instead of the search button to bring up the search page.

As seen in left image in Figure 4, the search tab allows the user to list all Individuals, all Head of Households (HOH), all Households (HH), or all MapNotes. Additionally, the user can search for a person by a maximum of four fields. By clicking on the combo box arrow and choosing a field name, the user can choose the field by which he or she wants to search. Then he or she must enter a value for the field name. Finally, he or she must click on the search button at the bottom of that page. All the search results will appear in the fourth page labeled “Results”. From there, the user can choose which Individual, Household, or MapNote he or she wants to view. The user can also choose to delete the selected Individual, Household, or MapNote.
Figure 4. The Search Page. The third page in the Introduction, the search page allows the user to find existing Individuals, Head of Households (HOH's), Households, or MapNotes. As shown by the left image, the first part of the page contains buttons which will list all existing Individuals, HOH's, Households, or MapNotes. The right image shows how the user can search for a specific individual by entering different field names and their corresponding values. All results are then shown the fourth “Results” page.

Figure 5. The Household section. This section contains three pages: H1, H2, and HH. H1 and H2 contain information about one specific Household. The left image shows the first Household page (H1). The right image shows the third page (HH) which lists all existing Household members and allows the user to view an Individual, delete an Individual, or attach a new Individual to the Household.
2.6.3 Household Section

As seen in Figure 5, the Household Section contains three pages. The first two pages (HI and H2) contain information about the Household. The third page (HH) shows a list of all the Individuals who are members of that Household.

The information in H1 and H2 corresponds to the paper form entitled “NEON Household Visit Form”. The information collected on these pages includes data about the head of Household, street address, city, state and zip code of the Household, and the annual Household income. The first page (H1) also contains a button “Show location” which will bring up the Map highlighting the building containing the Household.

From the HH page, the user has a choice of viewing the selected Individual, deleting the selected Individual, or attaching a new Individual to the Household (see Figure 5). In the list of existing Household members, a “(h)” besides the name indicates that that Individual is the head of Household.

2.6.3 Individual Section

As seen in Figure 6, the Individual Section contains six pages: I1, I2, I3, Sum, Ref, and Rel. The first three pages I1, I2, and I3 contain information about one specific Individual. Corresponding to the two-paged paper forms entitled “NEON Individual Survey and Assessment Form”, these three pages include data such as last name, first name, current employment, education, health issues, and insurance coverage.
The fourth page (Sum) contains another two sub-pages: Options and Sum. The first page (Options) shows a list of all the Summaries associated with that Individual. The user can choose to view a selected Summary, delete a selected Summary, or attach a new Summary. When the user chooses to view a selected Summary, the second page (Sum) appears showing the selected Summary’s information. Similarly, if the user chooses to attach a new Summary, the second page also appears, except this time it is blank so that the user can fill in the new Summary information. The fifth (Ref) and sixth (Rel) pages are identical to the Sum page except they display Referral or Release Information. These two pages also have a button for Client or Staff Signatures. Clicking on this button will bring up a Signature Capture window, which allows the user to sign his or her name in a box (see Figure 7). The signature is then saved. The next time the user clicks on the signature button for that specific Referral or Release form, the saved signature will appear in the Signature Capture window.
Figure 6. The Individual section. This section contains five pages: I1, I2, I3, Sum, Ref, and Rel. I1, I2, and I3 contain information about one specific Individual. The Sum page allows the user to add, delete, or view Summary forms; the Ref page allows the user to add, delete, or view Referral forms; and the Rel page allows the user to add, delete, or view Release forms. The Sum, Ref, and Rel pages each have two sub-pages. The first page (Options) shows a list of all the existing Summaries, Referrals, or Releases. The second page (Sum, Ref, or Rel) either shows the data of an existing Summary, Referral, or Release. Or it is a blank form so that the user can fill in the appropriate data. The left image shows I1, while the right image shows the Summary page and its two sub-pages.

Figure 7. Signature Capture. This window allows the user to sign his or her name in the box, and the signature will be captured. The user can save the signature, clear the signature, or close the window. The Signature Capture window is used in the Referral (Ref) and Release (Rel) pages of the Individual Section.
On the bottom of the Map is a toolbox. The reset button will zoom out the map to the full extent seen in the left image of Figure 8. Using the first checkbox group, the user can choose either to navigate the map by zooming or panning. Using the second checkbox group, the user can choose either view Households or MapNotes. When the user first clicks on a building, the building shape turns red. If the user clicks again on the building, a window will show up listing either all the Households connected to that building or all the MapNotes connected with that building, depending on what the user has selected in the second checkbox group (see Figure 8). The save button will the save any changes made to the map and, the clear button will clear all selections (red streets or red buildings) from the map.

Figure 9. Households list. This figure shows the list of Households that appears after the user double-clicks on a building. From the list of Households, the user can choose to view the selected Household, add a new Household to the building, connect an existing Household to the building, or to cancel and close the window.
On the bottom of the Map is a toolbox. The reset button will zoom out the map to the full extent seen in the left image of Figure 8. Using the first checkbox group, the user can choose either to navigate the map by zooming or panning. Using the second checkbox group, the user can choose either view Households or MapNotes. When the user first clicks on a building, the building shape turns red. If the user clicks again on the building, a window will show up listing either all the Households connected to that building or all the MapNotes connected with that building, depending on what the user has selected in the second checkbox group (see Figure 8). The save button will the save any changes made to the map and, the clear button will clear all selections (red streets or red buildings) from the map.

Figure 9. Households list. This figure shows the list of Households that appears after the user double-clicks on a building. From the list of Households, the user can choose to view the selected Household, add a new Household to the building, connect an existing Household to the building, or to cancel and close the window.
When the list of Households appears after the user double-clicks on a building, he or she can choose to view the selected Household, add a new Household to the building, or connect an existing Household to the building (see Figure 9). If the user chooses to view a selected Household or add a new Household, the first page of the Household section will be displayed. If the user chooses to connect an existing Household, another window will pop up with a list of existing Households. The user can then select one of those Households to connect to the building. If the selected Household is already attached to a different building, a dialog window will pop up, warning the user that the Household will be attached to a new building and the old attachment will be deleted.

Similarly, when the list of MapNotes appears after the user double-clicks on a building, the user can choose to view the selected MapNote or add a new MapNote. In both cases, a new MapNotes window will appear. It will either contain existing MapNote data, or it will be blank allowing the user to enter a new MapNote. Figure 10 shows how a MapNote contains a subject, text, a date, and the name of the staff member who created the MapNote. A toolbar at the bottom of the allows the user either to save the MapNote, cancel and close the window, view on the Map the building to which the MapNote is connected, or to delete the MapNote.
3. IMPLEMENTATION (Results)

FieldJournal is implemented based on the Model-View-Controller (MVC) architecture. The Model classes store all the data and perform the calculations for the creating the Map. The View classes, which make up the user interface, contain no data; they are simply empty user interface frames. Linking the Model and the View, the Controller classes manage all the events and fill the View’s user interface frames with the appropriate data from the Model. In this implementation, the Model does not know that the View exists and never directly sends the View any data. Instead, after the Controller has updated the Model, the Controller notifies the View to get the appropriate data from the Model classes.
3.1 Model

The Model contains two main groups of classes. The first group is the GeoTools package of classes which was adapted to run on FieldJournal specifications (J2ME, CDC, and Personal Profile). No major changes were made to GeoTools. Only some of the code was modified to accommodate for the functions that this J2ME platform does not have in its API.

The second group of classes is the data collection classes. They include a Record class, a Table class, a Household class, a Person class, and a MapNotes class. The Record is the most basic class and simply stores the data hash table. There are six types of Records: Person, Household, MapNotes, Summary, Referral, and Release. The type of the Record
depends on the kind of data it holds—A Household-type Record contains Household data, an Individual-type Record contains Individual data, etc. The Household class contains one instance of a Household-type Record; the MapNotes class contains one instance of a MapNotes-type Record; the Person class contains one instance of a Person-type Record and various instances of Summary-, Referral-, or Release-type Records. In a Record, each data field is assigned an integer, and the data is later retrieved from the hash table with this integer. For example, for an Individual, the last name field is assigned the integer ‘3’. To retrieve a specific Individual’s last name, the hash table in the Individual’s Record is referenced with the integer ‘3’.

There is exactly one Table for every running instance of FieldJournal. The Table class points to all the instances of the Household class. In addition to containing one Household-type Record, the Household class points to all the Person instances representing the members of the household. As described above, each Person instance contains many Record instances—exactly one Person-type Record and zero or more Summary-, Referral-, or Release-type Records.

3.2 View

The View contains all the user interface classes. These classes initialize and layout the widgets, panels, and windows with which the user interacts. These classes do not contain any data but simply point to the appropriate instances of a Household, Person, MapNotes, or a Record. The View classes use widgets from the Zaval Light-weight Toolkit. Although, the Zaval Light-weight Toolkit has a general event manager that allows the
widget to catch and process various events (e.g. button clicks, mouse movements, etc.), in FieldJournal, none of the widgets perform the actual handling of the events. They just pass the event to the Controller classes.

3.3 Controller

The Controller classes handle all the event management. Whenever the user interacts with a View class, the Controller processes the event, and sends the event to the Model to update the data. After the data has been updated, the Model notifies the Controller, which then tells the View class to get the updated data from the Model. The Controller consists of two main classes, FieldJournal and FJViewer. FieldJournal handles the event management associated with data collection. In order to take some of the work from FieldJournal, FJViewer handles the event management for the mapping part. If the user changes the view of the map by zooming, panning, or resetting, the FJViewer catches that event and notifies the Model. The Model will perform the necessary calculations to update the Map; this includes updating an off-screen image of the Map. After this is done, the Controller notifies the View that an update is ready, and the View gets the updated Map image from the Model and paints it on-screen.

3.4 Signature Capture

Because some of the forms (Referral and Release of Information) require signatures, a Signature Capture feature is included in FieldJournal. The Signature Capture differs from the rest of FieldJournal in that it does not use the Zaval Light-weight Toolkit for the user interface. Instead, it uses pure Java AWT widgets. The Signature Capture feature
pops up a new window with a box where the user can sign. It then saves the signature both as a text file (a list of points) and a PNG image file. When the user wants to view a previous signature in FieldJournal, the signature is reconstructed from the list of points in the text file. An open-source PNG encoder class written by David Eisenberg was used to encode the Java image created from the Signature Capture into a PNG image (Eisenberg, 2003). This PNG image is saved to the PDA and can later be used to re-create the signatures outside of FieldJournal.

4. DESIGN CHOICES (Discussion)

4.1 Use of J2ME, CDC, and Personal Profile
The J2ME platform is comprised of different configurations and profiles. Each combination of configurations and profiles was designed and optimized for specific devices. The Connected Device Configuration (CDC) and the Personal Profile combination, on which FieldJournal runs, are designed for devices with larger memory footprints, such as higher-end PDAs. The other J2ME combination consists of the Connected, Limited Device Configuration (CLDC) and the Mobile Information Device Profile (MIDP) is designed for devices with smaller memory footprints, such as mobile phones (Sun Microsystems, 2004). Currently, the CLDC and MIDP platform runs on many brands of mobile phones, and the most popular PDA operating systems (Palm OS and Pocket PC) support CLDC 1.0 and MIDP 1.0. Because the CLDC 1.0 and MIDP 1.0 combination can run on so many more devices, I wanted to design FieldJournal to run on that J2ME platform. This would have allowed for much more flexibility. However,
CLDC 1.0 and MIDP 1.0 do not support doubles or floats, so creating maps would have been nearly impossible. I researched ways to get around this, such as storing all doubles and floats as strings. However, it would require extra memory to perform the necessary conversions to handle doubles and floats. The memory constraint was already very tight (32KB for the runtime heap), so using extra memory seemed wasteful.

I then turned to CLDC 1.1 and MIDP 2.0 which would support doubles and floats. Originally, I did not want to use this platform because no PDAs or mobile phones have yet to support it. I did not want to risk designing FieldJournal for a platform which would never be supported. However, since it would be impossible to design FieldJournal without doubles and floats, and as device makers promised to release CLDC 1.1 and MIDP 2.0 support soon, I decided to start working with CLDC 1.1 and MIDP 2.0. (A Java Virtual Machine support CLDC 1.1 and MIDP 2.0 was released in March 2004. (PalmOne, Inc., 2004. para. 3.)) I created a simple, prototype program which could display, pan, and zoom on maps (see Figure 12). The main problem with CLDC 1.1 and MIDP 2.0 was that, being designed for mobile phones, the user interface widgets were very primitive, only simple textboxes and lists. I had to create my own button by drawing a rectangle and testing if the mouse pointer was within the rectangle’s bounds. Additionally, despite what device the program runs on, using CLDC 1.1 and MIDP 2.0, the program would always look like it was running on a mobile phone. At this time, I decided that if FieldJournal were going to be usable for outreach workers on PDAs, I would need to use a different platform.
Next, I decided that restricting FieldJournal to run only on PocketPCs would be a fair trade-off if that meant FieldJournal could be more usable. I explored using ESRI’s ArcPad, the mobile version of ESRI’s mapping software that runs only on PocketPCs. ArcPad also has a VBScript API so programmers could customize their maps, write scripts, or develop extension programs (ESRI, 2004). Using ArcPad to handle all the mapping, I intended to write VBScripts to handle the data collection. However, I soon realized that ArcPad was only a mapping application, and the VBScript API could not handle large amounts of data collection. First of all, the VBScript API did not offer any form widgets, such as textboxes or comboboxes. Secondly, ArcPad was very inflexible to work with—ArcPad had a mapping object model, and everything had to revolve around that model. This became an unnecessary constraint on the design of FieldJournal. Thus, in the end, the ArcPad API was not extensive or flexible enough. It could only handle mapping functions and could not accommodate the data collection process. Restricted to ArcPad’s limited widgets, FieldJournal would not have been very usable.
After experimenting with ArcPad, I decided to return to J2ME but this time using the CDC and Personal Profile configuration that only runs on PocketPCs. I originally did not consider this option because I wanted FieldJournal to be able to run on all PDA platforms. However, that condition was too restricting. Through the different prototypes created, I learned that FieldJournal would not be usable unless it only ran on a single PDA platform. So in the end, usability won over flexibility. FieldJournal users would have to use PocketPCs, but they would have more much enjoyable experience!
4.2 Use of Model-View-Controller Architecture

In the implementation of FieldJournal, I used the Model-View-Controller (MVC) architecture to optimize the program. Because memory is limited on a PDA, any little adjustment to the program made a big difference. The separation of the Model and View is the first such optimization, saving both time and space. Since the View classes are just empty frames, it takes less time to load them and less space to store them. For example, when the user clicks on the first Household page, the Household frame already exists, and the data for that Household already exists. The View simply has to fill that frame with the data. While multiple Household datasets may exist (one dataset for each Household), only one Household frame exists for the entire FieldJournal program. Thus, both time and space are conserved. On the other hand, if the Model and View were combined, either more time or more space would be required. With the Household, for example, Household user interface frame would have to be stored with each Household dataset. In this case, storing the Household data would require almost twice the amount of space. To save space, the Household data could be stored without the user interface frame. Then, each time the Household page loads, a new frame has to be created. This would double the time needed to load the page. Thus, without the Model-View separation, either more space or more time is required.

The Controller is separated from the Model and View so that one central class can act as a dispatch center and handle all the events. Without the Controller, the View classes would have to catch all events and send them to the appropriate Model classes. However, each View class only knows of the Model classes in which the updated data is obtained.
This keeps the size of the View classes small. However, often, events will need to be processed by Model classes about which the View class does not know. Without the Controller class, each View class would have to know about all the Model classes. This would increase the size of the View classes. Thus, in order to effectively separate the View and Model classes and keep the size of the View classes small, a Controller needs to act as the dispatch center, catching events from the View classes and fielding them to the appropriate Model classes.

4.3 Zaval Light-weight Toolkit

Although use of the Zaval Light-weight toolkit slowed down the program considerably (see Section 4.4 below). I decided that the more complex widgets that it offered outweighed the speed issue. Without the Zaval Light-weight toolkit, I would not have been able to use the tabs or the comboboxes. Those two widgets are central to making FieldJournal easier and faster to use for the user.

4.4 Signature Capture

At first I tried using the Zaval Light-weight toolkit to create the Signature Capture feature. However, it was too slow because the Zaval toolkit adds an extra layer to the event handling hierarchy. Because the Zaval Frame class is a subclass of the Java AWT Frame, each mouse movement would be caught by a Java AWT Frame and the Java AWT event manager then sent through the Zaval toolkit, which would then send the event to the appropriate component. In most of FieldJournal, this slowdown was an acceptable trade-off for the complex widgets that the Zaval toolkit offered (see Section
4.3) However, in the Signature Capture program, this slowdown could not be tolerated. In capturing signatures, the more points the program can catch, the more smooth and authentic the signature looks. With the Zaval toolkit, about 40 points were caught for my signature; without the Zaval toolkit and using just the Java AWT widgets, about 200 points were caught. The \( \frac{1}{5} \) less points caught by the Zaval toolkit made the signature unrecognizable and invalid. Thus, the Zaval toolkit could not be used to create the Signature Capture feature. Instead, the Java AWT toolkit was used.

4.5 Global Variables Class

Accessed by both the Model classes and the Controller classes, the Global Variables class is another small optimization. This class is only used when the Controller needs to send a variable to a GeoTools class in the Model. In the GeoTools model, that variable would have to be sent down through several layers to GeoTools classes before it would be used. In most cases, that variable would be sent down, but never actually used. Only in a few rare occasions would the variable be used. So, instead of sending down a useless variable through many layers of GeoTools classes, I put these variables into a static Global Variables class. Thus, if the GeoTools class needs to use one of those variables, it can access it directly through the Global Variables class.
4.6 New Households or Individuals

To create new Households or Individuals, I considered two options:

1. When the user wants to add a new Household or Individual, the program could create an empty Household or Individual and add a pointer to that Household or Individual in the backend table. During the next save, the modified table is written to a file.

2. When the user wants to add a new Household or Individual, the program would not add the pointer to that new Household or Individual in the backend table until the next save. During the next save, the new pointer is added to the table, and the modified table is written to a file.

The two options basically differ in when to add the new Household or Individual to the table, either during the creation of the Household or Individual or during the next save. Problems arise because FieldJournal periodically auto-saves all the data. So the user has no control over when the next save will happen.

With the first option of adding the pointer when the Household or Individual is first created, the program is essentially adding an empty Household or Individual to the table. If the user decides to create a new Household or Individual then changes his or her mind, the empty Household or Individual has already been saved to the table. The second option, adding the Household or Individual to the table at save-time, can avoid this problem. However, with the second option, each time a save occurs, FieldJournal would have to check if the new Household is empty (and thus not add a pointer to the table) or
not. This would slow down the program and create an unnecessary check for non-empty Households.

In the end, I decided that the first option was the more efficient one. At worst, an empty Household is saved. The user can later go back and delete it. The second option slows down the program unnecessarily because it is a rarer that the user would create a new Household or Individual and change his or her mind and not fill in any data. It is more common that the user would create a new Household or Individual and fill in the data. Thus, it would be a waste of time to perform a check for empty Households or Individuals during each save.

4.7 Saving to a XML file

When a Table is saved to a file, it is written in a XML format (see Figure 14 in Appendix E for a sample Table output file). Each object (Record, Person, Household, MapNotes, and Table) has a string representation of itself. The string representation of a Record is a tab-delimited list of all the data values in that Record, enclosed by the tags “<Record> and </Record>”. The string representation of a Person is a list of the Person-type Record and all the Summary-, Referral-, and Release-type Records belonging to that Person, all enclosed by the tags “<Person>” and “</Person>”. The string representation of a Household is a list of the string representations of all the Persons in that Household and the string representation of the Household-type Record, all enclosed by the tags “<Household>” and ”</Household>”. The string representation of a MapNotes is the MapNotes-type Record, enclosed by the tags “<MapNotes>” and “</MapNotes>”.

45
string representation of the Table is a list of string representations of all the Households and MapNotes in that Table, enclosed by the tags “<Table>” and “</Table>”. To save the Table to a file, the string representation of the Table is simply written to a file.

The XML format was used because there are so many different embedded objects involved—a Table contains Household objects, a Household contains a Record and Person objects, and a Person object contains many different types of Records. Writing in a XML-format was the best way to handle these embedded objects. Incorporating an XML-parser into the code was also much more straightforward than writing new code to handle reading a Table file. Finally, a table written in a XML-format is easier to export to a database program, which could then use a XML parser to extract out the data.

4.8 Notebook Model

In order to help the users of FieldJournal adapt to using the program, the interface had to be modeled after an already existing paradigm. The only constraint on the model was that it had to include some type of navigation menu. Because there are so many different sections and components to FieldJournal, the user needs this navigation menu to show him or her all the options available to him or her. I considered two models:

1. The Web Model. FieldJournal would act like a website. Each section of FieldJournal would be a webpage. A “Back” button would allow the user to return to the previous page. Acting as the navigation menu, a row of the hyperlinked text would run across the top of each page.
2. The Notebook Model. FieldJournal would imitate a notebook. Each section of FieldJournal would be a section of the notebook. The sections would be marked out by different tabs. A user could return to the previous section by clicking on the tab for that section. These tabs would also act as the navigation menu, informing the user of the different sections available to him or her.

I chose to model FieldJournal after a notebook because the outreach workers using this program are not as familiar with technology. They understand a notebook with its different tabs and sections more than they understand a website.

In the Notebook Model, within the different sections, if the user clicks on certain buttons, new pages in different sections will open up to the user. For example, in the Introduction page, if the user clicks on the “New Household” button, a page in the Household section will appear. Unlike the Web Model though, there is no back button. The user can go “back” only by selecting the tab for the section where he or she was. Often, the user cannot go even “back” to exactly where she was before. In the Results page of the Introduction section, once a Household, Individual, or MapNote has been selected, and the user has clicked on the “view” button, the Household, Individual, or MapNotes page will appear. At that point, the user cannot return to the previous Results page. However, the user can return to the Search page in the Introduction section, and all the search parameters the user had originally used will still be shown so the user can re-do the search. The Results page of the Introduction section is disabled after the user has left it.
because otherwise the information in it may be outdated. For example, if the Results page shows a list of Households and the user, after leaving the Results page, has deleted some of the Households, when he or she returns to the Results page, the information in the Results page will still include the just deleted Households. On the other hand, with the disabled Results page, the user returns to the Search page, re-does the search, and can view the updated information on the Results page.

In order to fully mimic a notebook, vertical tabs (tabs along the vertical edge) should have been used. However, vertical tabs would have taken up some of the width of the screen. Because the PDA screen is already limited in its width, using vertical tabs would force the use of a horizontal scrollbar. However, even without the use of any tabs, the user would already need to use a vertical scrollbar. Thus, the use of vertical tabs would mean the user would need a horizontal and vertical scrollbar. This would be too much scrolling. On the other hand, the use of horizontal tabs (tabs along the horizontal edge) would force the use of a vertical scrollbar. Since a vertical scrollbar was going to be needed anyhow, using horizontal tabs would not have any affect on the scrolling. So, with the use of horizontal tabs, FieldJournal looks less like a real notebook but allows the user to scroll less. This is a fair tradeoff.

4.9 Linked Fields

One of the features of FieldJournal is that once certain fields are filled in, they are automatically filled out in all other parts of FieldJournal. For example, once the Individual's name has been entered in the first page of the Individual section, it will
automatically be entered in the Referral pages. There were two main ways this could have been implemented: Either the linking would occur in the View, or the linking would occur in the Model.

If the linking occurred in the View, the textfield widgets would have to be linked. In this scenario, if the user changes one textfield, the other textfield would also change. The Model would never know about the linking; it would be purely a user interface feature. On the other hand, if the linking occurred in the Model, once the data for one field were entered, it would be stored in the in the backed Record multiple times (one time for each field that contained that data). The View would never know about the linking.

FieldJournal performs the linking in the Model. Once the user fills in the textfield in the View, the Controller gets the new data and sends it to the Model. Then, the Model links the data by storing it in the backend Record multiple times. Finally, the Controller notifies the View to get the updated data from the Model. Because the Model has stored the data in the backend Record multiple times, the data will appear multiple times in the View. So the next time the user comes across a textfield asking for previously entered information, that textfield will already be filled in.

However, the data is only linked in one direction, from the first textfield to the remainder of the textfields. If the user were to change a later textfield, it would not affect the other textfields. Also, the data is only linked the first time. Once the later textfields have been saved, they will no longer be linked to the first textfield. For example, when the user
creates a new Referral for an Individual, the first time the Referral page appears, the
name of the Individual will automatically be filled in from the data entered in the earlier
Individual section pages. However, the user can change the name in the Referral page.
This will not affect the name of the Individual in any other pages. Once the user saves
that Referral page, it will no longer be linked. If the user later changes the Individual’s
name in an earlier Individual section page, the name in the Referral will not change
because the Referral has already been saved. FieldJournal does this so that to allow full
flexibility for the user. The linked fields feature is only meant to be an interface
optimization for the user, so that he or she does not have fill in as many fields; it was not
meant to constrain the user. For example, if the user wanted to enter an Individual’s legal
name in the Individual data pages but wanted to enter the Individual’s nickname in the
Referral pages, he or she could not do this if the name fields were always linked bi-
directionally.

Because the linked fields feature is just an interface optimization for the user, it should
have been implemented in the View. Implementing it in the Model implies that the
linked fields feature is an intrinsic trait of data collection instead of just an interface
optimization for the user. However, performing the linking in the View would have been
much more complicated and slower than performing it in the Model. So, even though,
theoretically, the linked field feature should have been implemented in the View, it was
implemented in the Model so as to not slow down the program.
4.10 Mapping Features

The Map shows 3 different layers—the NEON zones, the NEON streets, and the NEON buildings. This is permanent and cannot be changed by the user. I did not allow the user to change the map layers because I felt it added unnecessary complications to the program. Realistically, the NEON outreach workers only need to see these three layers.

I had also hoped to put orthographic pictures in the Map to make it look more graphically appealing. However, orthographic pictures required too much memory. It was not worth the using the extra memory just to make the map look better.

I also considered putting street names next to the NEON streets. However, this would have cluttered the Map too much and taken away from the main functionality of the Map. Instead, I chose to allow the user to click on the street, and then the street name would show up in the upper left hand corner. Although it takes more effort on the part of the user, it is a cleaner way to display street names.

4.11 Feature Attribute Update

Originally, I wanted to include a Map feature attribute update function. This would allow the users to click on a Map object and view the spatial attributes found in the Map's DBF file. It would also allow users to add and edit spatial attributes. However, looking at the DBF files for the three different map layers, I found that there were not many interesting attributes to view or edit, only the generic attributes such as shape id or shape area which the user should not change. So instead, I designed the MapNotes function to be
extensible to adding new attributes. Since each MapNote object is already connected to a
Map object (in this case, a building), the MapNote text can easily be extended to being
attribute data. I included a sample function which would search all MapNotes for a
certain subject line as specified by the user. For every MapNote that matches the subject
line, the function extracts the text of that MapNote. Then the function creates a new DBF
file, which is a copy of the old DBF file with an extra column appended to it. The extra
column header is the subject line and the data in that column is the MapNotes text
corresponding to each Map object. This sample function is not part of the actual
FieldJournal application because, at this point, the NEON outreach workers do not need
to update attributes on a PDA. However, in the future, this sample function and more
based upon it could be part of a desktop computer application that accompanies
FieldJournal.

4.12 Testing

The original design procedure called for three iterations of testing with the NEON
outreach workers. I had hoped they would give me abundant feedback about the design
of FieldJournal and its usability. However, the NEON outreach workers did not have
much feedback to give me. They mainly commented on minor details, such as what data
to put in the combobox lists, not large design or usability issues. Since I wanted to
ensure that the NEON outreach workers felt involved in the design process of the
FieldJournal, I proceeded through the three iterations of testing, but FieldJournal did not
change much through those iterations. The testing procedure consisted of two steps:
First, I would give a general explanation of the program or any changes I had made to the
program. Then, each NEON outreach worker would take turns and proceed through the normal process of collecting data on a Household and Individual.

4.13 Further Work - Accompanying Desktop Program

Originally, I had hoped to also write a desktop computer application which would synchronize the data from the PDA to the centralized database that resided in a desktop computer. However, for many reasons, doing this within the NEON context proved impossible at the current time. However, with the Table saved as a XML file, it should be easy to later write this desktop computer program, which would (1) synchronize data from the PDA with the data in the centralized database, (2) allow the user to extract specific data from the database to put into the PDA. The database must also be slightly altered to allow a field for the OBJECT_ID in the Household data section. The OBJECT_ID field is the crucial link between the collected non-spatial data and the spatial mapping datasets. FieldJournal automatically creates this link, but desktop computer application must preserve that link through the OBJECT_ID field.

I also included some sample functions which the desktop computer program could use. Two of these functions create HTML files out of the Referral and Release of Information data. These HTML files include the signature of the clients and residents obtained by FieldJournal (see Appendix B and C for a sample output of these functions). I also wrote two sample functions that use the Household data or MapNotes to create new map attributes. These sample functions show how easy it is to alter map datasets and create new maps based on the data collected in FieldJournal.
4.14 Evaluating the Study’s Success

The NEON outreach workers were very excited about using FieldJournal in their work. Throughout this study, I was intending for the outreach workers to begin using FieldJournal on PDAs by the fall of 2004. However, because funding for this project was withdrawn at the last minute, the database synchronizing program could not be written, and I was unable to obtain enough PDAs for the outreach workers. Therefore, they will not be using it in the near future. In that sense, the study failed the hard criteria for success (producing a product that would be used). However, based on the feedback I received during testing sessions, I believe that if the outreach workers could obtain enough PDAs, and if the database synchronizing program could be written, they would indeed use FieldJournal. So, even though the complete study was not a success, but the part that was completed, the PDA application, was a success!

5. CONCLUSION

FieldJournal was designed to investigate the integration of data collection and mapping in a seamless software application in the context of the outreach workers at NEON. Learning about the problems NEON faced in data collection and mapping, from a computer scientist’s point of view, I thought the solution was easy—combine data collection and mapping and thus, remove the tedious work of the middle man! However, in building FieldJournal, I had to think a lot about the work processes of the NEON outreach workers. I soon realized that data collection is one process that is almost entirely separate from the mapping process. Data collection is done by all the outreach
workers in the field. The main motivation of data collection is to meet the residents of the North End neighborhood and help provide for their different needs. Mapping is done by a GIS expert in the office. The main motivation of mapping is to create maps to present to various organizations so that they will partner with NEON. In FieldJournal, it was hard to combine the two processes while still preserve the already established workflow of the outreach workers. For example, I wanted to add more mapping functions, but that disturbed the data collection workflow and added unnecessary complications. So, in the end, FieldJournal became a data collection program with an extra mapping feature. FieldJournal become more data-centric with occasional mapping. I do not feel this fully bridges the gap between data collection and mapping, but it is a start. The gap cannot be fully bridged until community planners such as the outreach workers at NEON view them as one process instead of two separate ones. The gap cannot be fully bridged until the outreach workers who collect the data also become the GIS experts who create the maps.

Although it was not one of the original purposes, FieldJournal also became an investigation into developing software for PDAs. During the study, I spent as much time exploring the new technologies being released for PDAs as I did designing FieldJournal. Building a software program for a PDA also presented many complications that could have been avoided in building a software program for a desktop computer. Many times, I was forced to sacrifice some functionality for speed. However, even as I was building FieldJournal, PDA technology was rapidly changing. New PDA processors were constantly being released, and memory was becoming cheaper and cheaper. Following this trend, the trade-offs I made in order to preserve the speed soon would not be needed.
On the other hand, I wonder if PDAs will even be used in 10 years. Laptops are becoming smaller and lighter, and, on the other end, mobile phones are becoming more and more advanced. Soon, laptops and mobile phones will meet and replace PDAs entirely. However, since FieldJournal was written in Java not some PDA operating system language, it will hopefully still be usable on any device that can run a Java Virtual Machine.

FieldJournal is an investigation into data collection and mapping process for community planners. It is also an investigation into developing software for PDAs. I hope that FieldJournal will be the start of something and not the end. This study has proven that data collection and mapping can be combined, but further work can be done into combining those processes more seamlessly. This study has proven the success of J2ME’s platform of the Connected Device Configuration and the Personal Profile on a PocketPC. However, further work should be done in spreading the attractiveness of that platform by building a Java CDC/Personal Profile Virtual Machine that can run on the PalmOS operating system. This study has proven the viability and usability of the Zaval Light-weight Toolkit, but further work should be done into optimizing that toolkit. Indeed, the FieldJournal study has proven that it is possible to build data collection and mapping software for community planners on PDAs. I hope it will be an inspiration for others to build better technology for community planners and for PDAs.
7. ACKNOWLEDGEMENTS

Special thanks to Raj Singh, who supervised this project; Ceasar MacDowell and Joy Amulya, who allowed me to do this work in their center; all the outreach workers at NEON for their input and enthusiasm, especially Julia Arroyo for acting as the main liaison between the NEON outreach workers and me; Sarah Williams for all the encouragement she gave me in the beginning; and finally and most importantly, the Waitt Institute for funding this project.

8. REFERENCES


9. APPENDIX

A. Table of Figures

B. FieldJournal’s Release of Information Form

C. FieldJournal’s Referral Form

D. FieldJournal User’s Guide

E. Technology Plan for NEON
APPENDIX A. Table of Figures

Figure 1. A symbolic representation of what FieldJournal does. ........................................21
Figure 2. Navigation between the different sections of the FieldJournal user interface. 23
Figure 3. The Introduction section. ................................................................. 24
Figure 4. The Search Page. ..................................................................................26
Figure 5. The Household section. ........................................................................26
Figure 6. The Individual section. .........................................................................29
Figure 7. Signature Capture. ................................................................................29
Figure 8. The Map section. ....................................................................................30
Figure 9. Households list. .....................................................................................31
Figure 10. A MapNote. .......................................................................................33
Figure 11. Implementation architecture and classes. ............................................34
Figure 12. CLDC 1.1 and MIDP 2.0 Prototype Program. .......................................39
Figure 13. ArcPad Prototype. ...............................................................................40
Figure 14. Example of the FieldJournal’s Output XML file. ...............................73
Table 15. Columns of Household Records. ..........................................................74
Table 16. Columns of Individual Records. ............................................................76
Table 17. Columns of Referral Records. ...............................................................76
Table 18. Columns of Release Records. .................................................................77
Table 19. Columns of Summary Records. .............................................................77
Table 20. Columns of MapNotes Records. ............................................................77
APPENDIX B. FieldJournal's Release of Information

(The form below was created from the HTML file that FieldJournal produces based on data collected.)

North End Outreach Network (NEON)
200 Birnie Avenue
Springfield, MA 01107
phone:(413)787-5191 fax: (413)787-5192

RELEASE OF INFORMATION

Client Name: Janet Lai   DOB: 3/10/1985
Address: 70 Pacific Street
City: Cambridge   State: MA   Zipcode: 02139

I, Janet Lai, give permission for the North End Outreach Network to talk, obtain from, or release information regarding me to RELEASE AGENCY.

I understand that all the information pertaining me will be strictly confidential by the North End Outreach Network. I also understand that this information will be kept confidential by the agency to which I am referred. I understand that this authorization to release information will be valid for one year and will automatically expire one year after it is signed. I understand that I may cancel this consent at any time without penalty.

I have read this information and have had it explained to me.

Client Signature: [Signature] Date: 07/10/2004

Staff Signature: [Signature] Staff Title: CHA   Date: 07/11/2004
APPENDIX C. FieldJournal’s Referral Form

(The form below was created from the HTML file that FieldJournal produces based on data collected.)

North End Outreach Network (NEON)
200 Birnie Avenue
Springfield, MA 01107
phone:(413)787-5191 fax: (413)787-5192

REFERRAL FORM

Date of Referral: 07/09/2004   Community Advocate Code: n001

Resident ID: LAJA10031981

Resident Name
Last: Lai  First: Janet  MI: W

Scheduled Appointment
Date: 07/15/2004   Time: 03:00 pm

Agency Name: Referral Agency
Contact Name: Jeff Chang
Agency Phone Number: (617)452-4219

Type of Referral: Health
Reason for Referral: testing

Resident Signature: [Signature]

CA Signature: [Signature]

Received by Referral Agency: ________________________________
APPENDIX D. FieldJournal User’s Guide

Installing the J9 on a PocketPC
1. Create the following file structure:
   - wsdd/
   - wsdd/bin
   - wsdd/app
   - wsdd/lib
   - wsdd/lib/jclFoundation
   - wsdd/lib/jclCdc
   - wsdd/lib/jclPPro
   - wsdd/data
2. Copy the following files into the wsdd/bin directory:
   - iverel2*.dll
   - j9.exe (console version of J9)
   - j9dyn2*.dll
   - j9prt2*.dll
   - j9thr2*.dll
   - j9vm2*.dll
   - j9w.exe (no-console version of J9)
   - j9zlib2*.dll
3. Copy the following files into the wsdd/lib directory:
   - jclFoundation/classes.zip
   - jclCdc/locale.zip
   - jclCdc/classes.zip
   - jclPPro/ppro-ui-win.zip
4. Execute the j9.exe by clicking on the icon in the bin directory to ensure that the J9 is properly installed.

Installing FieldJournal on a PocketPC
1. Install the J9. See Installing the J9 on a PocketPC.
2. Copy the FieldJournal.jxe into the wsdd/app folder.
3. Copy the FieldJournal.lnk into the wsdd/app folder and the /Windows/Start Menu/wsdd folder. You may need to create that latter folder.
4. Copy the following directories into the wsdd/data folder:
   - NEONzones.shp, NEONzones.dbf
   - northendbuildings.shp, northendbuildings.dbf
   - northendstreets.shp, northendstreets.dbf

Starting FieldJournal
1. Go to the Start menu and click on the “wsdd” folder.
2. Click on the “FieldJournal” icon. Note FieldJournal may take several minutes to start up. Also, the first time that FieldJournal starts up, an error window containing the message “Error loading table” may appear. Ignore this window. If the window pops up again, there are errors loading the data table.
Exiting FieldJournal
1. Save any changes by clicking on the “save” button at the bottom of the page.
2. Click on the “exit” button at the bottom of the page.

Adding or Viewing Household Information
1. Bring up the Household section by adding a new Household or viewing an already existing Household. See Adding a New Household or Finding an already existing Household.
2. View or fill out the two pages of information in the tabs labeled “H1” and “H2”.
3. To attach Individuals to this new Household, click on the tab labeled “HH”, and click on the “Attach New” button. The Individual Section will then appear.
4. To view an already existing Individual in this Household, click on the tab labeled “HH”. Select the already existing Individual, and click on the “View Selected” button.

Adding or Viewing Individual Information
1. Bring up the Individual section by adding a new Individual, attaching a new Individual to a Household, or viewing an already existing Individual. See Adding a New Individual or Finding an already existing Individual.
2. View or fill out the three pages of information in the tabs labeled “I1”, “I2”, and “I3”.
3. View or edit Summaries by clicking on the tab labeled “Sum”.
   a. The “Options” page that appears allows you to view an already existing Summary, delete an already existing Summary, or attach a new Summary.
   b. To view or delete an already existing Summary, select the Summary from the list in the left box. Then click on the “View Selected” button or the “Delete Selected” button.
   c. To attach a new Summary, click on the “Attach Selected” button and fill out the information in the “Sum” page that appears.
4. View or edit Referrals by clicking on the tab labeled “Ref”.
   a. The “Options” page that appears allows you to view an already existing Referral, delete an already existing Referral, or attach a new Referral.
   b. To view or delete an already existing Referral, select the Referral from the list in the left box. Then click on the “View Selected” button or the “Delete Selected” button.
   c. To attach a new Referral, click on the “Attach Selected” button and fill out the information in the “Ref” page that appears.
5. View or edit Release of Information forms by clicking on the tab labeled “Rel”.
   a. The “Options” page that appears allows you to view an already existing Release, delete an already existing Release, or attach a new Release.
   b. To view or delete an already existing Release, select the Release from the list in the left box. Then click on the “View Selected” button or the “Delete Selected” button.
   c. To attach a new Release, click on the “Attach Selected” button and fill out the information in the “Rel” page that appears.
Adding a new Household
There are two ways of doing this:
A. From the Intro section
   1. In the first row of tabs, click on the “1” tab.
   2. In the second row of tabs, click on the “Intro” tab to bring up the Intro page.
   3. Click on the button labeled “new HH”
   4. The Household section pages should then appear. See Adding or Viewing Household Information.

B. From the Map
   1. In the first row of tabs, click on the “M” tab.
   2. In the toolbar near the bottom of the page, click on the “hh” checkbox.
   3. Use the zoom and pan functions to find the building associated with the new Household.
   4. Click once on the building, and the building should turn red.
   5. Click on the building again to bring up the “Households” dialog window.
   6. Click on the “Add New HH here” button.
   7. The Household section pages should then appear. See Adding or Viewing Household Information.

Adding a Map location to an already existing Household
1. Note the address of the already existing Household
2. On the first row of tabs, click on the tab labeled “M”.
3. In the toolbar near the bottom of the page, click on the “hh” checkbox.
4. Use the zoom and pan functions to find the building associated with the new Household.
5. Click once on the building, and the building should turn red.
6. Click on the building again to bring up the “Households” dialog window.
7. Click on the “Connect Existing HH here” button.
8. A dialog window with a list of all the existing Households will appear. Select the Household from that list.
9. Click on the “Connect this HH here” button.
10. The Household should now be connected to that location. The Household information can be obtained by double-clicking on that building and selecting the Household from the list that appears. The location on the map can also be viewed by clicking on the button labeled “view on map” in the “H1” page of the Household section.

Finding an already existing Household
There are two ways of doing this:
A. From the Intro section
   1. Note the address of the already existing Household
   2. In the first row of tabs, click on the “1” tab.
   3. In the second row of tabs, click on the “Intro” tab to bring up the Intro page.
   4. Click on the “search” button or the tab labeled “Search”.
   5. In the “Search” page that appears, click on the “List all Households” button.
6. In the “Results” page that appears, select the Household, and click on the “View” button.
7. The Household section pages with the appropriate Household information will appear. See Adding or Viewing Household Information.

B. From the Map
1. Note the address of the already existing Household
2. In the first row of tabs, click on the “M” tab.
3. In the toolbar near the bottom of the page, click on the “hh” checkbox.
4. Use the zoom and pan functions to find the building associated with the new Household.
5. Click once on the building, and the building should turn red.
6. Click on the building again to bring up the “Households” dialog window.
7. Select the Household from the list at the top of the window, and click on the “View Selected HH” button.
8. The Household section pages with the appropriate Household information will appear. See Adding or Viewing Household Information.

Deleting an already existing Household
1. Note the address of the already existing Household
2. In the first row of tabs, click on the “1” tab.
3. In the second row of tabs, click on the “Intro” tab to bring up the Intro page.
4. Click on the “search” button or the tab labeled “Search”.
5. In the “Search” page that appears, click on the “List all Households” button.
6. In the “Results” page that appears, select the Household, and click on the “Delete” button.
7. In the dialog window that appears, click “yes” to delete the Household.

Adding a new Individual to an already existing Household
There are two ways of doing this:
A. From the Intro section
1. In the first row of tabs, click on the “1” tab.
2. In the second row of tabs, click on the “Intro” tab to bring up the Intro page.
3. Click on the button labeled “new Individual”.
4. A list of already existing Households will show up on the “HH” page. Select a Household from this list, and click on the “Create new individual” button.
5. The Individual section pages will then appear. See Adding or Viewing Individual Information.
B. From the Household section
1. Bring up the Household section by adding a new Household or viewing an already existing Household. See Finding an already existing Household.
2. To attach Individuals to this new Household, click on the tab labeled “HH”, and click on the “Attach New” button. The Individual Section will then appear. See Adding or Viewing Individual Information.
Adding a new Individual to a new Household
1. Create a new Household. See Adding a new Household.
2. Attach a new Individual to that new Household. See Adding a new Individual to an already existing Household.

Finding an already existing Individual
There are three ways of doing this:
A. From the Intro Section – List All
   1. In the first row of tabs, click on the “I” tab.
   2. In the second row of tabs, click on the “Intro” tab to bring up the Intro page.
   3. Click on the “search” button or the tab labeled “Search”.
   4. In the “Search” page that appears, click on the “List all Individuals” button.
   5. In the “Results” page that appears, select the Individual, and click on the “View” button.
   6. The Individual section pages with the appropriate Individual information will appear. See Adding or Viewing Individual Information.
B. From the Intro Section – Search parameters
   1. In the first row of tabs, click on the “I” tab.
   2. In the second row of tabs, click on the “Intro” tab to bring up the Intro page.
   3. Click on the “search” button or the tab labeled “Search”.
   4. In the “Search” page that appears, scroll to the bottom half of the page.
   5. Fill in the search parameters. There are four sets of two boxes. You can use one, two, three, or four of those sets. In each of those sets, first choose a search field by clicking on the down arrow next to the top box. Choose the appropriate field.
   6. Fill in the search value in the lower box. Repeat for different search fields.
   7. Click on the “search” button at the bottom of the page.
   8. In the “Results” page that appears, select the Individual, and click on the “View” button.
   9. The Individual section pages with the appropriate Individual information will appear. See Adding or Viewing Individual Information.
C. From the Household Section
   1. Bring up the Household section by adding a new Household or viewing an already existing Household.
   2. To Individuals belonging to this Household, click on the tab labeled “HH”. Select an Individual from the list, and click on the “View Selected” button. The Individual Section will then appear. See Adding or Viewing Individual Information.

Deleting an already existing Individual
There are two ways of doing this:
A. From the Intro section
   1. In the first row of tabs, click on the “I” tab.
   2. In the second row of tabs, click on the “Intro” tab to bring up the Intro page.
   3. Click on the “search” button or the tab labeled “Search”.
   4. In the “Search” page that appears, click on the “List all Individuals” button.
5. In the “Results” page that appears, select the Individual, and click on the “Delete” button.

6. In the dialog window that appears, click “yes” if you are sure you want to delete the Individual.

B. From the Household section
   1. Bring up the Household section for the Household associated with this Individual. See Finding an already existing Household.
   2. In the Household section, click on the third tab labeled “HH”.
   3. Select the Individual from the list on the left side, and click on the “Delete Selected” button.
   4. In the dialog window that appears, click “yes” if you are sure you want to delete the Individual.

Adding a new MapNote
   1. In the first row of tabs, click on the “M” tab.
   2. In the toolbar near the bottom of the page, click on the “notes” checkbox.
   3. Use the zoom and pan functions to find the building associated with the new MapNote.
   4. Click once on the building, and the building should turn red.
   5. Click on the building again to bring up the “MapNotes” dialog window.
   6. Click on the “Attach new MapNote” button.
   7. A MapNotes window will appear. Fill in the appropriate data. The toolbar at the bottom of the MapNotes allows you to save the MapNote, cancel, view the building associated with this MapNote on the Map, or to delete the MapNote.

Viewing an already existing MapNote
There are two ways to do this.
A. From the Intro section
   1. In the first row of tabs, click on the “1” tab.
   2. In the second row of tabs, click on the “Intro” tab to bring up the Intro page.
   3. Click on the “search” button or the tab labeled “Search”.
   4. In the “Search” page that appears, click on the “List all MapNotes” button.
   5. In the “Results” page that appears, select the MapNote, and click on the “View” button.
   6. A MapNotes window will appear with the appropriate data. The toolbar at the bottom of the MapNotes allows you to save the MapNote, cancel, view the building associated with this MapNote on the Map, or to delete the MapNote.

B. From the Map
   1. In the first row of tabs, click on the “M” tab.
   2. In the toolbar near the bottom of the page, click on the “notes” checkbox.
   3. Use the zoom and pan functions to find the building associated with the new MapNote.
   4. Click once on the building, and the building should turn red.
   5. Click on the building again to bring up the “MapNotes” dialog window.
   6. Select a MapNote from the list, and click on the “View selected MapNote” button.
7. A MapNotes window will appear. Fill in the appropriate data. The toolbar at the bottom of the MapNotes allows you to save the MapNote, cancel, view the building associated with this MapNote on the Map, or to delete the MapNote.

Deleting an already existing MapNote
1. In the first row of tabs, click on the “1” tab.
2. In the second row of tabs, click on the “Intro” tab to bring up the Intro page.
3. Click on the “search” button or the tab labeled “Search”.
4. In the “Search” page that appears, click on the “List all MapNotes” button.
5. In the “Results” page that appears, select the MapNote, and click on the “Delete” button.
6. In the dialog window that appears, click “yes” if you are sure you want to delete the MapNote.
APPENDIX E. Technology Plan for NEON: Integrating FieldJournal into the Data Collection and Mapping System

FieldJournal is only a data collection and mapping application for Pocket PC-based Personal Digital Assistants (PDAs). Further work needs to be completed before FieldJournal can be integrated into a complete data collection and mapping system. This document describes what FieldJournal provides and the further work that needs to be done in order to create a complete system.

What FieldJournal provides

FieldJournal outputs all the data into one XML file. Each object (Record, Person, Household, MapNotes, and Table) has a string representation of itself. The string representation of a Record is a tab-delimited list of all the data values in that Record, enclosed by the tags “<Record> and </Record>”. The string representation of a Person is a list of the Person-type Record and all the Summary-, Referral-, and Release-type Records belonging to that Person, all enclosed by the tags “<Person>” and “</Person>”. The string representation of a Household is a list of the string representations of all the Persons in that Household and the string representation of the Household-type Record, all enclosed by the tags “<Household>” and “</Household>”. The string representation of a MapNotes is the MapNotes-type Record, enclosed by the tags “<MapNotes>” and “</MapNotes>”. The string representation of the Table is a list of string representations of all the Households and MapNotes in that Table, enclosed by the tags “<Table>” and “</Table>”. This Table is saved into the XML file. Figure 14 shows an example of this XML file. The Table contains one Household with the street address of ‘70 Pacific
Street’. The Household contains one Individual, ‘Janet W Lai’, who is the head of the Household. This Individual has one Summary, one Release, and Referral.

The tab-delimited lists are always ordered in the same way so that each string between tabs represents a value in a column. Tables 15-20 show the column names associated with each column. In the example in Figure 14, the first string in the Household’s Record is the date, the second string is the CAC (Community Advocate Code), etc.

**Further Work Needed**

Based on the structure of the XML file, a database synchronization program needs to be created. This program will have four main features:

1. This program will use the XML file to update NEON’s centralized database (residing in a desktop computer) with the new data.

2. The program also needs to allow the user to extract specific data from the centralized database and put that data in the PDA. The program must create a file identical to FieldJournal’s output XML file so that FieldJournal can access the data.

3. The program will allow the user to create HTML versions of the Referral and Release of Information forms. This will allow the users to print paper copies of those forms. Sample functions are included in FieldJournal.

4. The program will allow the user to update exist map attributes by editing the DBF file associated with map layers. The program should be able to edit existing map attributes as well as add new ones. The map attributes will be
added or edited based on Household, Individual, or MapNotes data. Sample functions are included in FieldJournal.

This program can be written in any programming language. However, since NEON’s current database is in Microsoft Access, the database synchronization program will most likely be written in Visual Basics.

**Technology Plan for NEON**

NEON will need to accomplish three things before the data collection and mapping system will be complete. First, NEON needs to purchase enough Pocket PC PDAs for its outreach workers. Second, NEON needs to find someone to write the database synchronization program. Third, NEON needs to hire someone to maintain the system. This person should be on call during the work days to help the outreach workers whenever they encounter problems. This person should be proficient in Java and in whatever language the database synchronization program is written. Initially, this person will spend most of his or her time fixing bugs that occur. As the system stabilizes, the person will spend more time maintaining the database and backing up the system.
Figure 14. Example of the FieldJournal's Output XML file. This file shows a Table with one Household with the street address of 70 Pacific Street. The Household contains one member, Janet W Lai, who is the head of the Household. This Individual contains one Summary, one Release, and one Referral. Note that extra line breaks in this file were taken out and the data was slightly modified, but the structure of the XML file remains the same.
<table>
<thead>
<tr>
<th>Household Records - Field Name</th>
<th>Column #</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>0</td>
</tr>
<tr>
<td>CAC</td>
<td>1</td>
</tr>
<tr>
<td>householdInactive</td>
<td>2</td>
</tr>
<tr>
<td>DOB</td>
<td>3</td>
</tr>
<tr>
<td>lastName</td>
<td>4</td>
</tr>
<tr>
<td>firstName</td>
<td>5</td>
</tr>
<tr>
<td>MI</td>
<td>6</td>
</tr>
<tr>
<td>ID</td>
<td>7</td>
</tr>
<tr>
<td>noIndAttached</td>
<td>8</td>
</tr>
<tr>
<td>zone</td>
<td>9</td>
</tr>
<tr>
<td>residenceCode</td>
<td>10</td>
</tr>
<tr>
<td>address</td>
<td>11</td>
</tr>
<tr>
<td>city</td>
<td>12</td>
</tr>
<tr>
<td>state</td>
<td>13</td>
</tr>
<tr>
<td>zipCode</td>
<td>14</td>
</tr>
<tr>
<td>home1978</td>
<td>15</td>
</tr>
<tr>
<td>weather</td>
<td>16</td>
</tr>
<tr>
<td>lat</td>
<td>17</td>
</tr>
<tr>
<td>lng</td>
<td>18</td>
</tr>
<tr>
<td>incEarnings</td>
<td>19</td>
</tr>
<tr>
<td>incChild</td>
<td>20</td>
</tr>
<tr>
<td>incSSI</td>
<td>21</td>
</tr>
<tr>
<td>incSSDI</td>
<td>22</td>
</tr>
<tr>
<td>incSS</td>
<td>23</td>
</tr>
<tr>
<td>incNonSS</td>
<td>24</td>
</tr>
<tr>
<td>incOther</td>
<td>25</td>
</tr>
<tr>
<td>incOtherSrc</td>
<td>26</td>
</tr>
<tr>
<td>incTotal</td>
<td>27</td>
</tr>
<tr>
<td>incRefused</td>
<td>28</td>
</tr>
<tr>
<td>incNone</td>
<td>29</td>
</tr>
<tr>
<td>TAFDAC</td>
<td>30</td>
</tr>
<tr>
<td>TAFDACdate</td>
<td>31</td>
</tr>
<tr>
<td>foodStamps</td>
<td>32</td>
</tr>
<tr>
<td>WIC</td>
<td>33</td>
</tr>
<tr>
<td>fuel</td>
<td>34</td>
</tr>
<tr>
<td>otherStateAid</td>
<td>35</td>
</tr>
<tr>
<td>otherStateAidSrc</td>
<td>36</td>
</tr>
<tr>
<td>incarcerated</td>
<td>37</td>
</tr>
<tr>
<td>computer</td>
<td>38</td>
</tr>
<tr>
<td>incEarningsMoYr</td>
<td>39</td>
</tr>
<tr>
<td>incChildMoYr</td>
<td>40</td>
</tr>
<tr>
<td>incSSIMoYr</td>
<td>41</td>
</tr>
<tr>
<td>incSSDIMoYr</td>
<td>42</td>
</tr>
<tr>
<td>incSSMoYr</td>
<td>43</td>
</tr>
<tr>
<td>incNonSSMoYr</td>
<td>44</td>
</tr>
<tr>
<td>incOtherMoYr</td>
<td>45</td>
</tr>
<tr>
<td>ObjectID</td>
<td>46</td>
</tr>
</tbody>
</table>

Table 15. Columns of Household Records. This table shows the column names associated with the columns in the Household-typed Records.
<table>
<thead>
<tr>
<th>Individual Records - Field Name</th>
<th>Column #</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>0</td>
</tr>
<tr>
<td>CAC</td>
<td>1</td>
</tr>
<tr>
<td>DOB</td>
<td>2</td>
</tr>
<tr>
<td>lastName</td>
<td>3</td>
</tr>
<tr>
<td>firstName</td>
<td>4</td>
</tr>
<tr>
<td>MI</td>
<td>5</td>
</tr>
<tr>
<td>ID</td>
<td>6</td>
</tr>
<tr>
<td>HHID</td>
<td>7</td>
</tr>
<tr>
<td>newHouse</td>
<td>8</td>
</tr>
<tr>
<td>relationHOH</td>
<td>9</td>
</tr>
<tr>
<td>idMom</td>
<td>10</td>
</tr>
<tr>
<td>idDad</td>
<td>11</td>
</tr>
<tr>
<td>SSN</td>
<td>12</td>
</tr>
<tr>
<td>status</td>
<td>13</td>
</tr>
<tr>
<td>email</td>
<td>14</td>
</tr>
<tr>
<td>homePhone</td>
<td>15</td>
</tr>
<tr>
<td>workPhone</td>
<td>16</td>
</tr>
<tr>
<td>marital</td>
<td>17</td>
</tr>
<tr>
<td>gender</td>
<td>18</td>
</tr>
<tr>
<td>ethnicity</td>
<td>19</td>
</tr>
<tr>
<td>nationality</td>
<td>20</td>
</tr>
<tr>
<td>language</td>
<td>21</td>
</tr>
<tr>
<td>speaksEng</td>
<td>22</td>
</tr>
<tr>
<td>rwEng</td>
<td>23</td>
</tr>
<tr>
<td>speaksSpan</td>
<td>24</td>
</tr>
<tr>
<td>rwSpan</td>
<td>25</td>
</tr>
<tr>
<td>empStatus</td>
<td>26</td>
</tr>
<tr>
<td>empName</td>
<td>27</td>
</tr>
<tr>
<td>jobTitle</td>
<td>28</td>
</tr>
<tr>
<td>farmWorker</td>
<td>29</td>
</tr>
<tr>
<td>NEWWorker</td>
<td>30</td>
</tr>
<tr>
<td>BayStateWorker</td>
<td>31</td>
</tr>
<tr>
<td>highGrade</td>
<td>32</td>
</tr>
<tr>
<td>school</td>
<td>33</td>
</tr>
<tr>
<td>specialEd</td>
<td>34</td>
</tr>
<tr>
<td>choiceSchool</td>
<td>35</td>
</tr>
<tr>
<td>bilingual</td>
<td>36</td>
</tr>
<tr>
<td>schoolAge</td>
<td>37</td>
</tr>
<tr>
<td>daycare</td>
<td>38</td>
</tr>
<tr>
<td>headstart</td>
<td>39</td>
</tr>
<tr>
<td>preschool</td>
<td>40</td>
</tr>
<tr>
<td>yearUSA</td>
<td>41</td>
</tr>
<tr>
<td>noYearsComm</td>
<td>42</td>
</tr>
<tr>
<td>birthPlace</td>
<td>43</td>
</tr>
<tr>
<td>primaryIns</td>
<td>44</td>
</tr>
<tr>
<td>secondaryIns</td>
<td>45</td>
</tr>
<tr>
<td>pcpName</td>
<td>46</td>
</tr>
<tr>
<td>pcpType</td>
<td>47</td>
</tr>
<tr>
<td>asthma</td>
<td>48</td>
</tr>
<tr>
<td>breastCancer</td>
<td>49</td>
</tr>
<tr>
<td>skin</td>
<td>50</td>
</tr>
</tbody>
</table>
Table 16. Columns of Individual Records. This table shows the column names associated with the columns in the Individual-typed Records.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Column #</th>
</tr>
</thead>
<tbody>
<tr>
<td>diabetes</td>
<td>51</td>
</tr>
<tr>
<td>heart</td>
<td>52</td>
</tr>
<tr>
<td>bloodPressure</td>
<td>53</td>
</tr>
<tr>
<td>learning</td>
<td>54</td>
</tr>
<tr>
<td>prostate</td>
<td>55</td>
</tr>
<tr>
<td>respiratory</td>
<td>56</td>
</tr>
<tr>
<td>tobacco</td>
<td>57</td>
</tr>
<tr>
<td>std</td>
<td>58</td>
</tr>
<tr>
<td>alcohol</td>
<td>59</td>
</tr>
<tr>
<td>drug</td>
<td>60</td>
</tr>
<tr>
<td>hiv</td>
<td>61</td>
</tr>
<tr>
<td>leadDone</td>
<td>62</td>
</tr>
<tr>
<td>leadHigh</td>
<td>63</td>
</tr>
<tr>
<td>leadTreat</td>
<td>64</td>
</tr>
<tr>
<td>depression</td>
<td>65</td>
</tr>
<tr>
<td>arthritis</td>
<td>66</td>
</tr>
<tr>
<td>other</td>
<td>67</td>
</tr>
<tr>
<td>otherSrc</td>
<td>68</td>
</tr>
<tr>
<td>immunized</td>
<td>69</td>
</tr>
<tr>
<td>address</td>
<td>70</td>
</tr>
<tr>
<td>city</td>
<td>71</td>
</tr>
<tr>
<td>state</td>
<td>72</td>
</tr>
<tr>
<td>zipCode</td>
<td>73</td>
</tr>
</tbody>
</table>

Table 17. Columns of Referral Records. This table shows the column names associated with the columns in the Referral-typed Records.

<table>
<thead>
<tr>
<th>Referral Records - Field Name</th>
<th>Column #</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>0</td>
</tr>
<tr>
<td>type</td>
<td>1</td>
</tr>
<tr>
<td>agencyName</td>
<td>2</td>
</tr>
<tr>
<td>contactName</td>
<td>3</td>
</tr>
<tr>
<td>phoneNo</td>
<td>4</td>
</tr>
<tr>
<td>dateAppt</td>
<td>5</td>
</tr>
<tr>
<td>timeAppt</td>
<td>6</td>
</tr>
<tr>
<td>reason</td>
<td>7</td>
</tr>
<tr>
<td>signClient</td>
<td>8</td>
</tr>
<tr>
<td>signClientLoc</td>
<td>9</td>
</tr>
<tr>
<td>signStaff</td>
<td>10</td>
</tr>
<tr>
<td>signStaffLoc</td>
<td>11</td>
</tr>
<tr>
<td>signReceived</td>
<td>12</td>
</tr>
<tr>
<td>timeAMPM</td>
<td>13</td>
</tr>
<tr>
<td>CAC</td>
<td>14</td>
</tr>
<tr>
<td>lastName</td>
<td>15</td>
</tr>
<tr>
<td>firstName</td>
<td>16</td>
</tr>
<tr>
<td>MI</td>
<td>17</td>
</tr>
<tr>
<td>ID</td>
<td>18</td>
</tr>
<tr>
<td>Release Records - Field Name</td>
<td>Column #</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>releaseAgency</td>
<td>0</td>
</tr>
<tr>
<td>dateReleaseClient</td>
<td>1</td>
</tr>
<tr>
<td>dateReleaseStaff</td>
<td>2</td>
</tr>
<tr>
<td>signStaff</td>
<td>3</td>
</tr>
<tr>
<td>signStaffLoc</td>
<td>4</td>
</tr>
<tr>
<td>releaseStaffTitle</td>
<td>5</td>
</tr>
<tr>
<td>signClient</td>
<td>6</td>
</tr>
<tr>
<td>signClientLoc</td>
<td>7</td>
</tr>
<tr>
<td>lastName</td>
<td>8</td>
</tr>
<tr>
<td>firstName</td>
<td>9</td>
</tr>
<tr>
<td>MI</td>
<td>10</td>
</tr>
<tr>
<td>DOB</td>
<td>11</td>
</tr>
<tr>
<td>address</td>
<td>12</td>
</tr>
<tr>
<td>city</td>
<td>13</td>
</tr>
<tr>
<td>State</td>
<td>14</td>
</tr>
<tr>
<td>zipCode</td>
<td>15</td>
</tr>
</tbody>
</table>

Table 18. Columns of Release Records. This table shows the column names associated with the columns in the Release-typed Records.

<table>
<thead>
<tr>
<th>Summary Records - Field Name</th>
<th>Column #</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>0</td>
</tr>
<tr>
<td>staffName</td>
<td>1</td>
</tr>
<tr>
<td>Needed</td>
<td>2</td>
</tr>
<tr>
<td>Provided</td>
<td>3</td>
</tr>
<tr>
<td>notes</td>
<td>4</td>
</tr>
<tr>
<td>lastName</td>
<td>5</td>
</tr>
<tr>
<td>firstName</td>
<td>6</td>
</tr>
<tr>
<td>MI</td>
<td>7</td>
</tr>
<tr>
<td>ID</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 19. Columns of Summary Records. This table shows the column names associated with the columns in the Summary-typed Records.

<table>
<thead>
<tr>
<th>MapNotes Records - Field Name</th>
<th>Column #</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>0</td>
</tr>
<tr>
<td>staffName</td>
<td>1</td>
</tr>
<tr>
<td>subject</td>
<td>2</td>
</tr>
<tr>
<td>notes</td>
<td>3</td>
</tr>
<tr>
<td>objectid</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 20. Columns of MapNotes Records. This table shows the column names associated with the columns in the MapNotes-typed Records.