Final Report

Mobile Geographic Information System (GIS) Solution for Pavement Condition Surveys

Florida Department of Transportation contract ***BDR76***

Prepared for:

Florida Department of Transportation, State Materials Office

Abdenour Nazef, P.E.
5007 NE 39th Avenue
Gainesville, FL 32609
(352) 955-6322

Prepared by:

Applied Research Associates

Transportation Sector
3605 Hartzdale Drive
Camp Hill, PA 17011
(717) 975-3550

June 2012

Disclaimer

The opinions, findings, and conclusions expressed in this publication are those of the authors and not necessarily those of the State of Florida Department of Transportation.

Unit Conversions

While this document does not spend very much time discussing data with units of measure, the following units are used either when describing stationing, hardware accuracy, or distress measurements.

US Customary	Multiply By	Metric
inch	25.4	millimeters
feet	0.305	meters
miles	1.61	kilometers
square feet	0.093	square meters
inches per mile (International Roughness Index)	0.0158	meters per kilometer (International Roughness Index)

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle	5. Report Date
Mobile Geographic Information System (GIS) Solution for Pavement Condition Surveys	June 28, 2012
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
Jacob Walter, P.E., Chadwick Becker, David Smelser	001008.00000.00000
9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
Applied Research Associates Transportation Sector 3605 Hartzdale Drive Camp Hill, PA 17011	11. Contract or Grant No. BDR76
12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
Florida Department of Transportation 605 Suwannee Street, MS 30 Tallahassee, FL 32399	Final, February 1, 2011 to June 30, 2012
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This report discusses the design and implementation of a software-based solution that will improve the data collection processes during the Pavement Condition Surveys (PCS) conducted by the State Materials Office (SMO) of the Florida Department of Transportation. This software replaces the prior method of Microsoft Excel spreadsheets with macro programming and integrates data from both the Department's Geographic Information System (GIS) and their legacy test section database. The software, known as the XPCS system, contains three components: a database of pavement condition, an office application used for planning and reporting, and a mobile application used for data collection and navigation.

17. Key Word	18. Distribution Statement
Pavement Condition, GIS, PCS, SMO	No restrictions.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	118	

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

Executive Summary

The State Materials Office (SMO) of the Florida Department of Transportation performs annual Pavement Condition Surveys (PCS) of the Department's pavement network. This work is performed by single person crews in a vehicle capable of measuring rutting, faulting, and ride quality (i.e., roughness) while traveling at prevailing traffic speeds. Information from these measurements along with visual evaluations and notations of pavement distresses are currently entered into a Microsoft Excel spreadsheet located in an onboard computer. The template for this spreadsheet (a separate file is used for each county) contains macros to perform basic error checking and integration of the automated and visual data.

There are several issues with the current approach. Among them are:

- A single operator is tasked with driving, navigating, data collection, and initial data verification
- A need to better plan and optimize routing and testing within a county
- Lack of geographic data available to the surveyor (e.g., cross streets, landmarks)
- Absence of progress reporting on a more regular basis (i.e., counties are recorded as either surveyed or not surveyed instead of tracking progress on a test section basis)
- Difficult to load resultant data into the Department's GIS for viewing or analysis

To address these issues, the Department, through the SMO, issued a Request for Proposals (RFP) titled "Mobile Geographic Information System (GIS) Solution for Pavement Condition Surveys". The intent of this RFP was to develop a computer-based solution that would both address the shortcomings of the current system while allowing better integration to the Enterprise GIS under development at the Department. The resultant software, XPCS, addresses many of these issues.

XPCS contains three components:

- Database – An independent database that is designed to contain all inspection data for the Department's test sections across multiple years. As this database uses existing test sections for its primary identification scheme, it is easily integrated back into the GIS.
- Office – A desktop application used for viewing and reporting designed to work with a live connection to the XPCS Database component of the system. This provides overall data, allows the user to explore specific data (e.g., inspections for a particular test section), and determine progress and overall network health through a reporting system. The office component also includes software that takes data from the mobile component and uploads that data to the Database.
- Mobile – An application used in the survey vehicles and designed so that a live connection to the XPCS Database component is not required.

This report discusses the methodology used to design and develop the software components of the XPCS system, the current state of those components, and suggestions for enhancement of the system in the future.

Table of Contents

List of Figures

Introduction

The Pavement Condition Survey (PCS) process is a well-established practice at the Florida Department of Transportation (FDOT) used to determine the overall condition of the FDOT pavement network and decide where to allocate resources (money, equipment, and personnel) related to the maintenance and rehabilitation of the pavement assets that belong to the agency. There are two different PCS surveys performed by the Department: project-level and network-level. Project-level surveys are used to measure specific pavement performance for rehabilitation design, project acceptance, warranty compliance, and research evaluation. These data tend to have significant detail to answer questions such as "What techniques will reduce wet weather accidents in this section?", "Does FDOT need to approach the contractor regarding warranty repair?" and "Should FDOT continue to use this specific rehabilitation technique?" Network-level surveys focus more on evaluating pavement performance for project prioritization and resource allocation across the entire State Highway System. Network-level surveys also tend to have less detail but cover a much greater surface area of pavement.

The focus of this project was to improve the network-level PCS process by incorporating software and GIS-based tools to improve the productivity of the surveyors, increase the accuracy of their surveys, reduce the amount of transit time between survey areas (*test sections*, in FDOT terminology), provide tools to supervisors to track progress and quality, and create reporting tools for supervisors to support the decision-makers at the Department.

To that end, the FDOT State Materials Office (SMO) selected Applied Research Associates (ARA) to assist them with the development and implementation of a new software system for the collection of network-level PCS data. This report discusses the process used to define the Department's needs, design the system, create the software components of the system, and implement the system at the SMO.

The result of this project is the Extended PCS (XPCS) software package. It consists of three main components:

- XPCS Database – This is the element of the system where all data is centrally stored and the data source upon which all reporting is performed.
- XPCS Office – This element of the system provides the in-office functions required of the software including the dashboard, supervisor review, data synchronization, routing, and reporting.
- XPCS Mobile – This element of the system is placed in the survey vehicles and is used by the field technical staff. It is designed to work in a disconnected environment where data is collected, verified, and stored locally. Upon returning to the office at the end of a work week, the field staff synchronizes the data collected and stored in their vehicle with the XPCS Database hosted in the office.

Methodology

Per ARA's proposal, the approach to this project has been in four phases as shown in Figure 1.

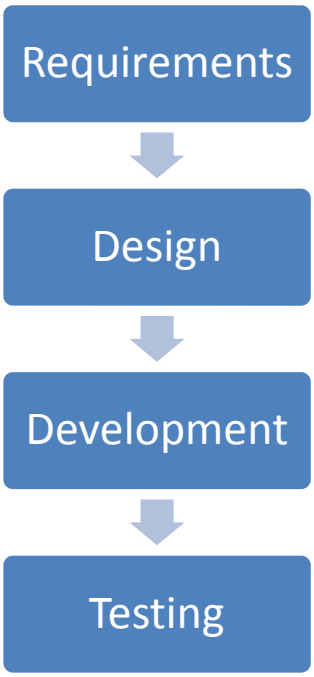

Figure 1. Project Approach

The first task was to examine the existing processes at the SMO related to the network-level PCS process. This task included interviews with SMO staff and ride-alongs with the SMO technicians responsible for the daily data collection. The ARA team also looked at existing data, process flow charts, and hardware involved in these efforts. ARA immediately followed up with the SMO and determined the desired improvements to the system such as status reporting, improved validation, and GIS integration. Based on these efforts, ARA began to draft and share versions of a pair of *Software Requirements Documents* (SRDs) with the SMO that discussed functional requirements (e.g., "Provide GIS Layer showing testing progress"), system configuration (the eventual three component system), and interface requirements (providing a visual representation of the initial versions of the final product for better understanding of the process). These drafts were reviewed and modified by the SMO and ARA based on feasibility, budget, existing FDOT systems, and required needs. The result was a pair of documents that could be checked to ensure that the final product met the requirements of the SMO and Department. The SRDs for both the office and mobile component of the system are attached to this report. The functions of the database component of the project are developed within these requirement documents as the visible components of this system are the office and mobile applications supported by the database.

Upon agreement of a final SRD for the project, the next task was to design the specific software that would meet the requirements set forth in the Requirements. This design would establish the technical specifics of the software such as database schema, internal object design, specific user interfaces,

required libraries, and interfaces with other systems such as the Roadway Characteristics Inventory (RCI) Database and GIS. The results of this process, a pair of *Software Design Documents* (SDDs) would provide the guidelines our developers would follow when writing the software. The SDD for both the office component and mobile component are also attached to this document.

The next phase of the work was the actual coding and integration of the various components of the new system. It was developed on a component-by-component basis. The first component developed was the XPCS Database. The database framework was created based on the SDD and then populated with data from the existing PCS dataset. This database was placed in the SMO offices where SMO staff could begin testing their own reporting. ARA also created some example reports to demonstrate the capabilities of the component.

The primary challenge with the data within the XPCS system is the creation and maintenance of location references. While FDOT maintains a canonical list of roadway segments in their RCI database, the inspection crews often break these areas up into smaller segments based on actual on-the-ground features (such as construction history). Furthermore, the boundaries of the entered inspection values were previously matched based on physical and geographical landmarks which could lead to errors depending on the description provided. Unfortunately, this precluded a database structure of natural keys (unique identifying fields that could describe a specific location) whereby segments (and their related historical inspection records) could be explicitly referenced by section id, roadway code, and beginning and ending mileposts. Additionally, new segment limits might be introduced in the field by the creation of breakpoints (possibly leading to conflicts during synchronization). The system was also complicated by the possibility of the roadway code changing from one year to the next because of a change from composite to divided roads.

To manage this issue, a system of segment IDs, tagged by the unique ID of the computer from which they were generated and a flag for travel direction (upstation or downstation) was introduced to address these synchronization issues. Also, a “5% rule” was developed to allow nearly, but not strictly, matching stretches of roadway to be associated with one another in the same manner as a person might, looking at the data. For example, if the data indicated that a segment existed on test section 26050000 between mileposts 8.05 and 9.20 but the surveyor observed a segment between mileposts 8.00 and 9.20, the program should treat that as the same segment and assign the new condition data to the original segment.

From here, a layer of business objects (XPCSData) was developed to represent the actual inspection data (and its relationship in the county->section->segment->inspection hierarchy) as shown in Figure 2.

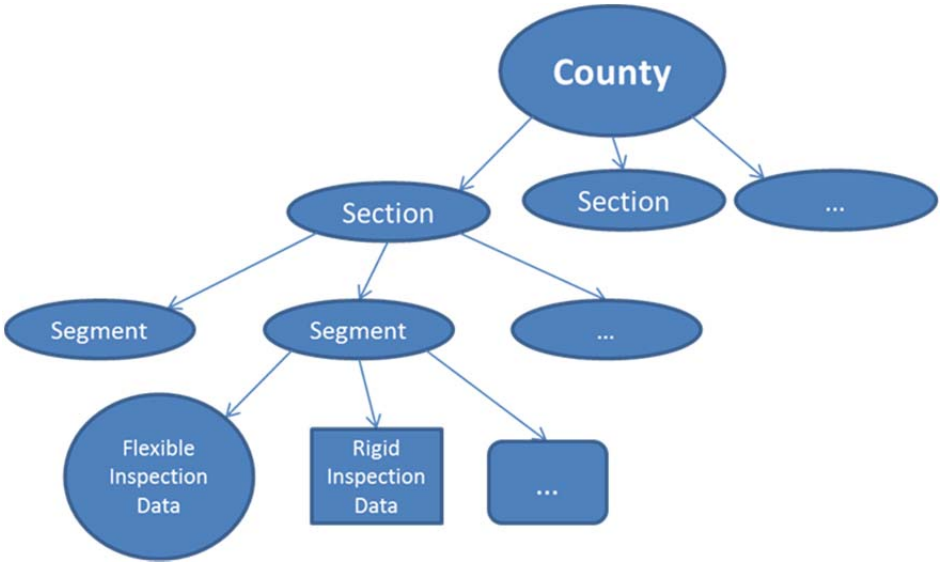

Figure 2. Layer of Business Objects

While in practical usage, rigid and flexible pavements are treated separately in the current PCS survey processes, there were many similarities that the developers were able to exploit. Most of the program operates on the more generic InspectionData object which could contain data regarding either flexible or rigid pavements. For those few operations that differ between the two pavement types and their condition data, two child classes were created: FlexibleInspectionData and RigidInspectionData. The advantage of this approach is that the code is more maintainable; most modifications simply need to be made to the InspectionData object. Only that code which performs operation unique to flexible or rigid pavement needs to be changed in the child objects.

The next component to be developed was XPCS Office. This component takes data from the XPCS Mobile component and transfers it to the XPCS Database. It also provides GIS services through the onboard SharpMap tools (an open-source GIS package) and the various functions defined in the SRD and SDD. The GIS system was designed with an eye for ease of use and simplicity. This design goal influenced everything from the function of the query tool (which avoids ambiguity when multiple sections may have been selected by zooming into the selection area) to the choice to invert the palette of colors (using a black background instead of the typical white). Additionally, the Office component was designed to provide flexible access to data by the linking of the GIS and Data Management tools. Selection of a section in either screen easily and quickly leads to its selection in the other.

Inspection data in the Data Management screen is automatically color coded to show the age of that data (current rows of data are white while previous data has a grey background which becomes darker depending on the age of the data) and is automatically grouped to let the user process the results in an intuitive manner. Other changes, such as the move from a “baked-in” dashboard (an interface that would need to be changed at the code level) to a window displaying one or more reports made the software easier to customize to a specific user or role (i.e., engineer, supervisor, and surveyor).

Finally, the XPCS Mobile component was developed and tested against the other two components. This software was tested by ARA developers and staff in the Champaign area (it was moved from the originally planned area in Pennsylvania to enable better debugging by the main development staff). Of course, the ARA IT environment and vehicle setup is not the same as that used by the SMO, so additional testing at the SMO was required.

The Mobile component was designed with flexibility in mind. Inspection crews are often forced to operate in sub-optimal traffic and construction environments and the ability to override the automatic section ordering provided by ArcGIS Network Analyst, the automatic screen flipping upon reaching the start and end of the test section, and the automatic adjustment of the zoom level of the MapPoint maps was deemed critical. While taking that need for flexibility into consideration, the Mobile component was designed to fully supplant the old Excel spreadsheet method of recording data while also providing an interactive map (updated in real-time) and direction to get the inspectors to the right place. The overall progress through the testing plan is shepherded by a constantly active Navigator object which manages the internal ordering of sections, the marking of sections as tested, and the automated management of the interface mode (navigation, testing, and verification).

Both components were designed with code-reuse in mind. The Office and Mobile applications use the same libraries to address data representation, form control (and even the same forms in some cases), validation, security, etc. In spite of a quite different look and feel, at every stage the underlying design principle was to treat the two components as two different ways of accessing and manipulating the same underlying system. In addition to the business logic layer (the code that handles activities such as validation), both applications use the same database structure. This simplifies the task of synchronization between the collection databases and the central database and eliminates the need to develop methods to convert information from a mobile format into an office format.

Deliverables

XPCS Database

The database is the core of the system and stores the data collected from the XPCS Mobile application for analysis and reporting by the XPCS Office application. It also stores FDOT-related data pulled from other systems within the Department such as the RCI database and the FDOT GIS basemap. This data is stored in a set of tables known as the Business Tables; the basic relationship of these tables is shown in Figure 3.

The database also stores all the security level data used to maintain data integrity and prevent data loss from non-administrative users. The basic relationships of these tables are shown in Figure 4.

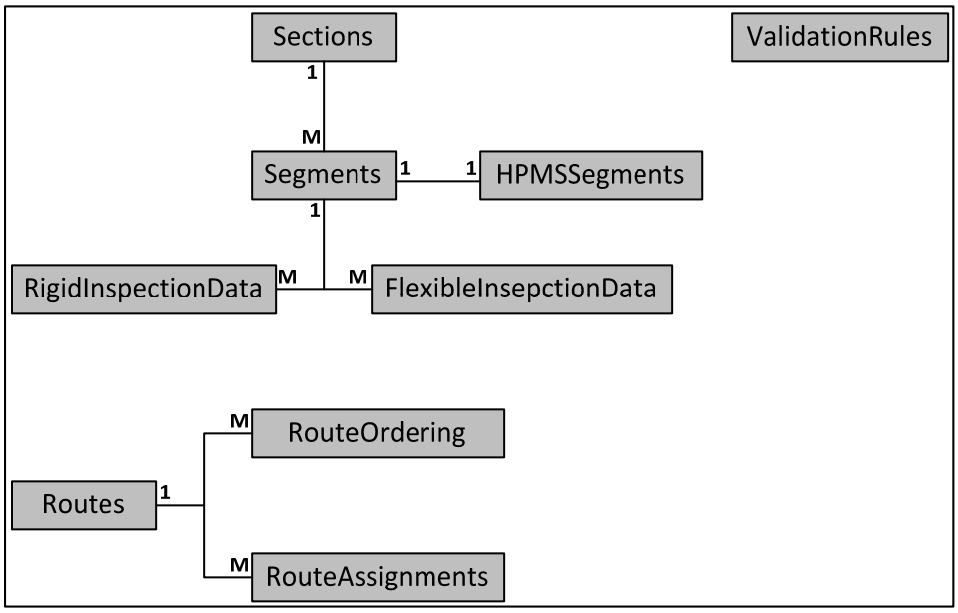

Figure 3. Business Tables

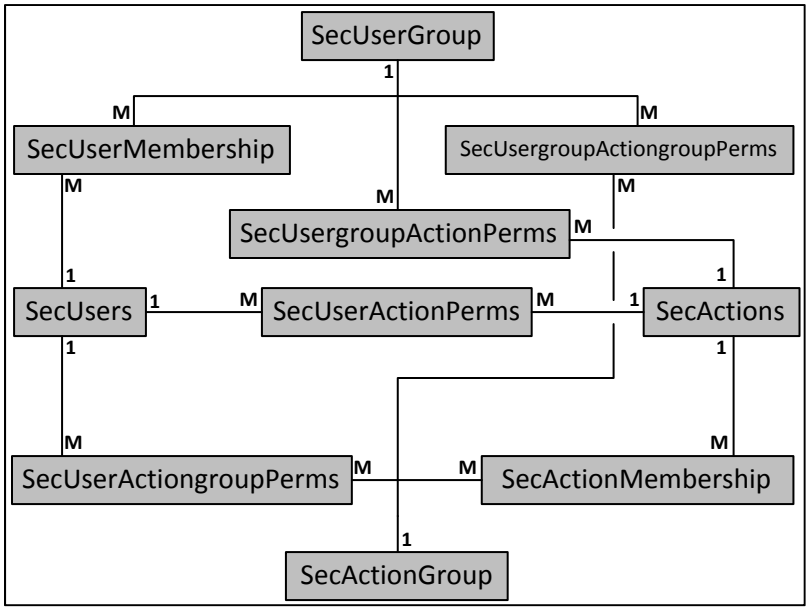

Figure 4. Security Tables

XPCS Office

The office component of the XPCS system is used for setup of data collection (i.e., route optimization), synchronization between the XPCS Database hosted within the FDOT IT infrastructure and the individual, disconnected XPCS Mobile field computers, validation, and reporting. It consists of the interfaces shown in Figure 5 through Figure 7. The layout is simple with the various functions represented by buttons on the left hand side and a large window on the right hand side storing GIS or various report results. Figure 5 shows the initial dashboard view that is displayed upon starting the software. The report that is generated may be specified by the user to match their needs and use case. In this example, a supervisor calls up a report on current progress across the statewide network when the software starts. Figure 6 shows the integrated GIS and data viewer elements of the system. This example shows in individual segment that has been selected in the GIS window and the condition data collected below. Darker rows in the grid represent older data. Finally, Figure 7 shows a sample report used for verification of data. This component can run any report on the database component of the software that the user has available. Reports are created with Crystal Reports 11.

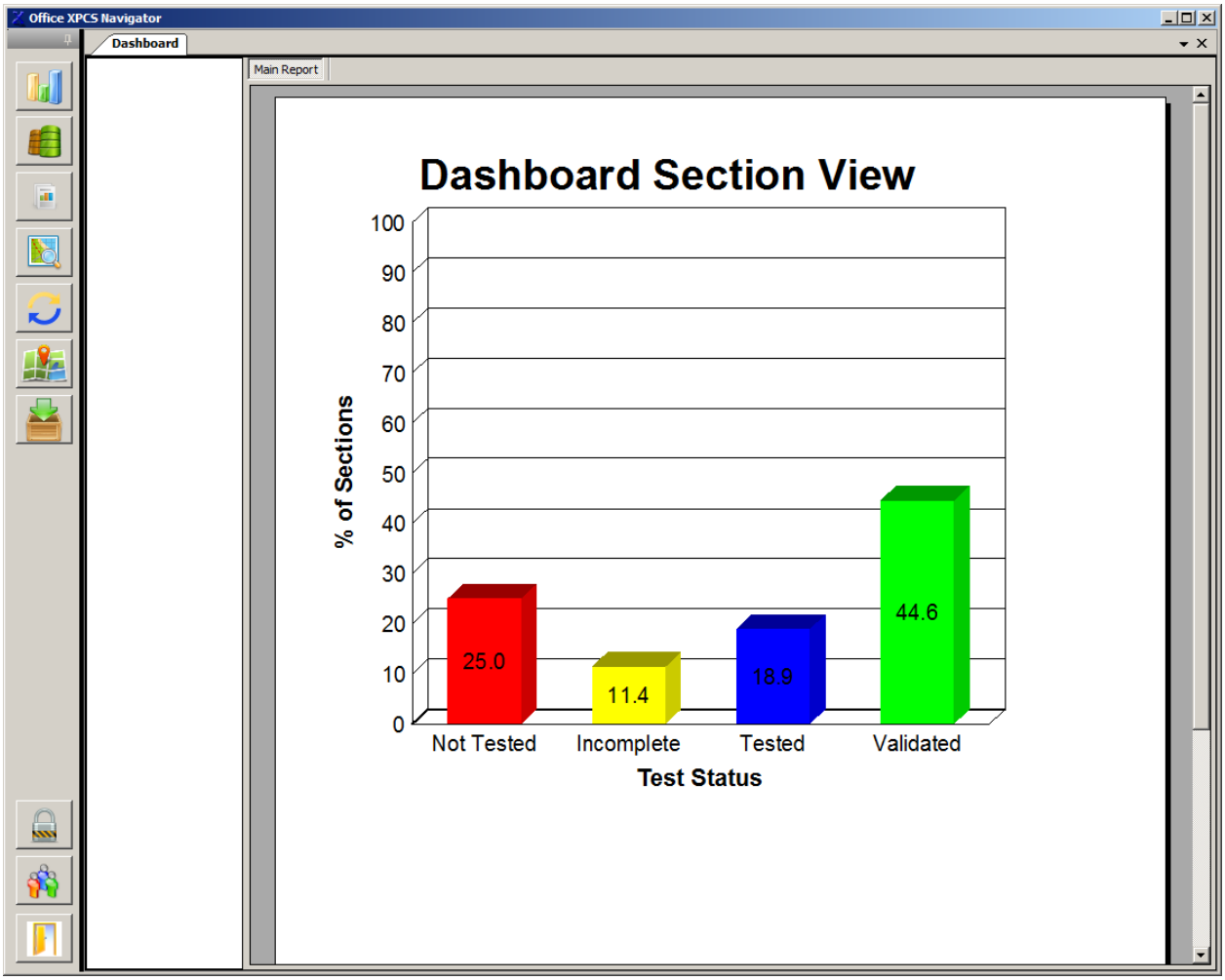

Figure 5. Dashboard Section View

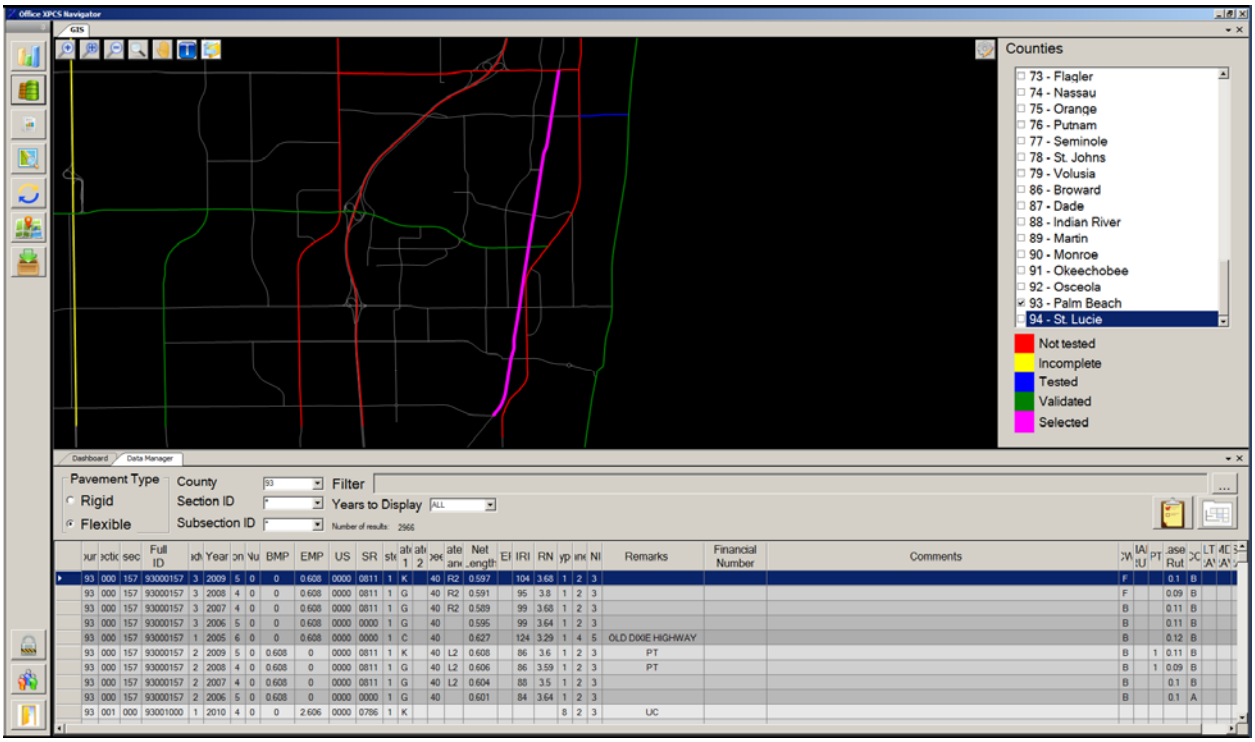

Figure 6. XPCS data viewing and integrated GIS

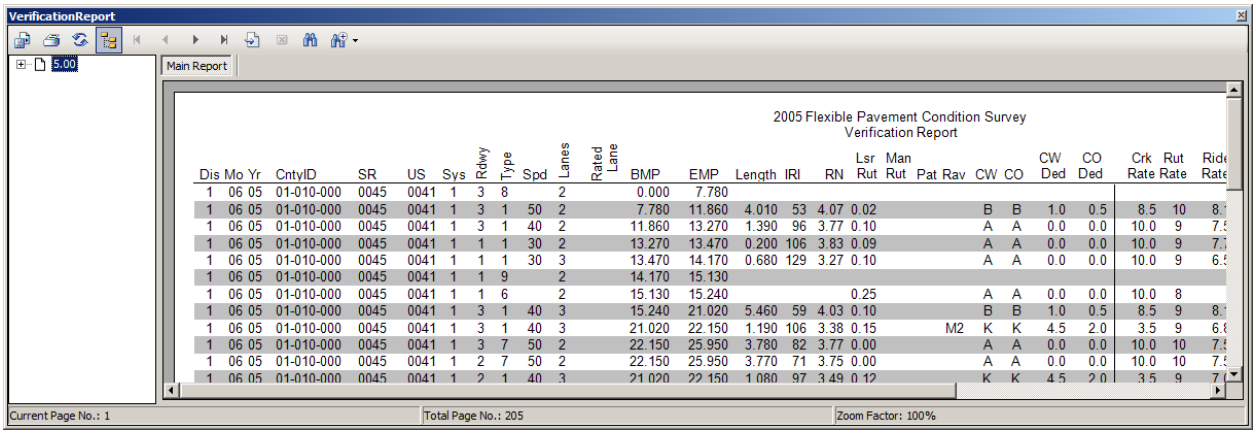

Figure 7. XPCS verification report

XPCS Mobile

The mobile component of the XPCS system is used in the field to collect data and perform immediate data validation. It operates in a disconnected environment; that is, data must be taken from the field computers and uploaded to the XPCS Database (the main data repository for this system) explicitly through the XPCS Office software.

The mobile component is broken up into three main sections.

The Section Navigator, shown in Figure 8, allows the operator to alter the order of sections to be visited as required and provides directions to the beginning of the test section. The surveyor may reorder the testing by dragging the individual elements of the route list. Route recalculation can also be invoked manually by clicking the Recalculate button in the lower right corner.

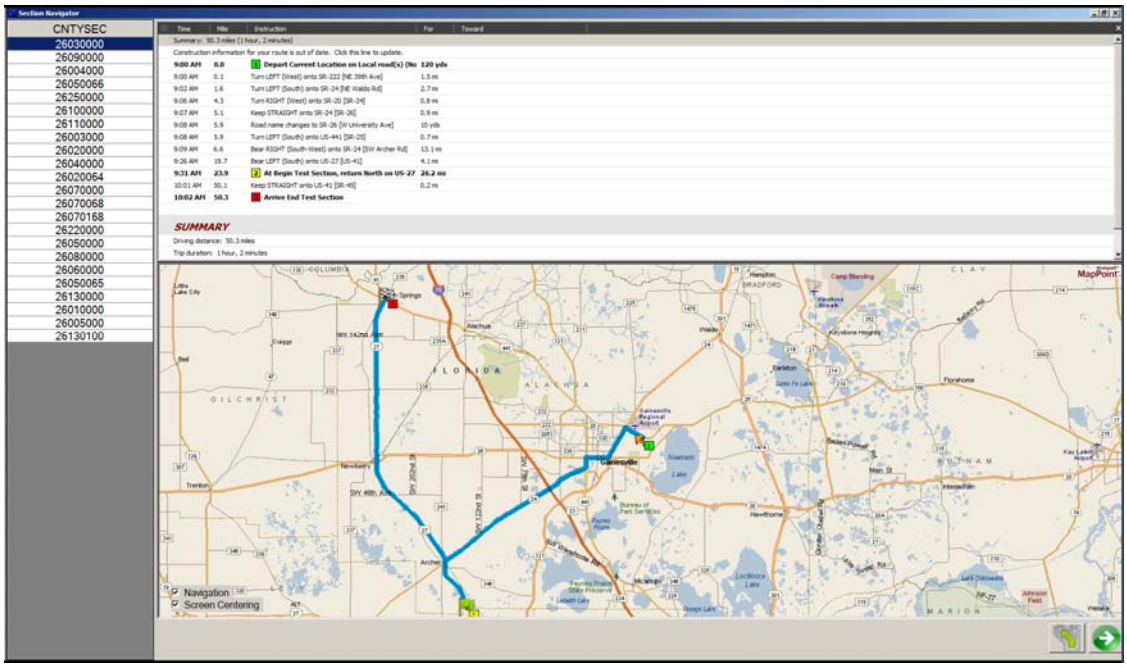

Figure 8. XPCS Mobile Section Navigator Screen

The Test Section interface, shown in Figure 9, guides the operator down the test section (or to the beginning of it, if they haven't yet reached it) and allows them to enter preliminary data while displaying the results of either last year's run or the previous run in the current testing cycle if a rerun is being performed. The surveyor can add breakpoints to create new segments within the test section through the boken chain button on the lower left of the interface.

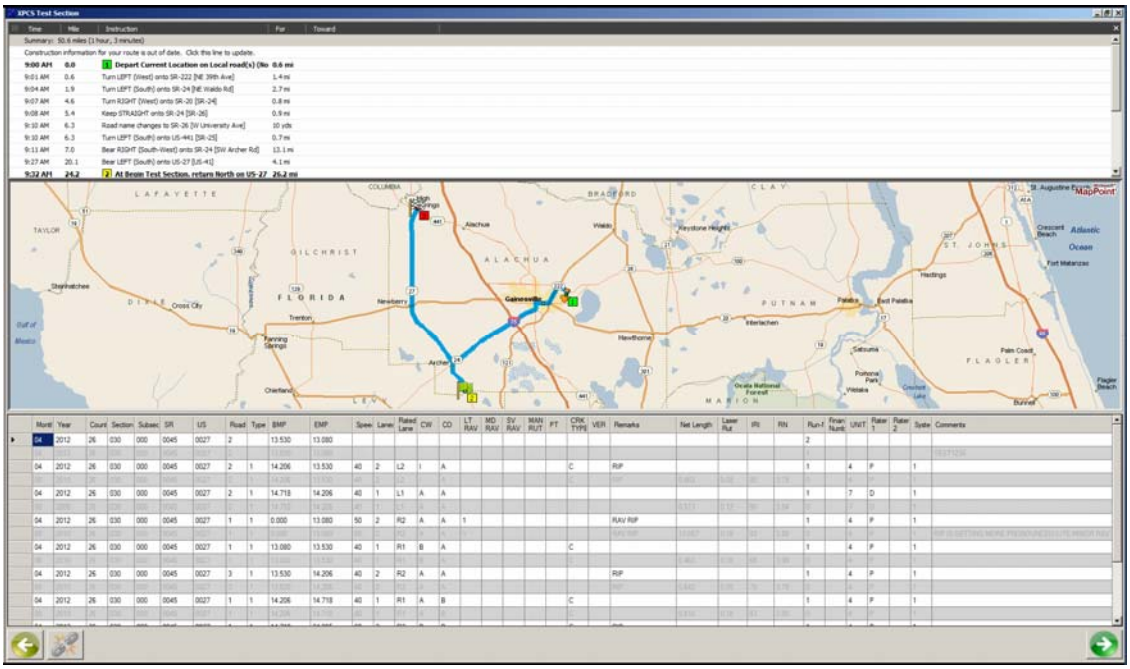

Figure 9. XPCS Mobile Field Test Screen

At the conclusion of a run, the vehicle operator will use the Data Validation screen (Figure 10). This interface is used to load data collected by WinRP as well as validate the collected/entered data. Prior year data is displayed in the grey rows between the active survey rows; the user can control the number of prior surveys shown through the drop-down box on the left.

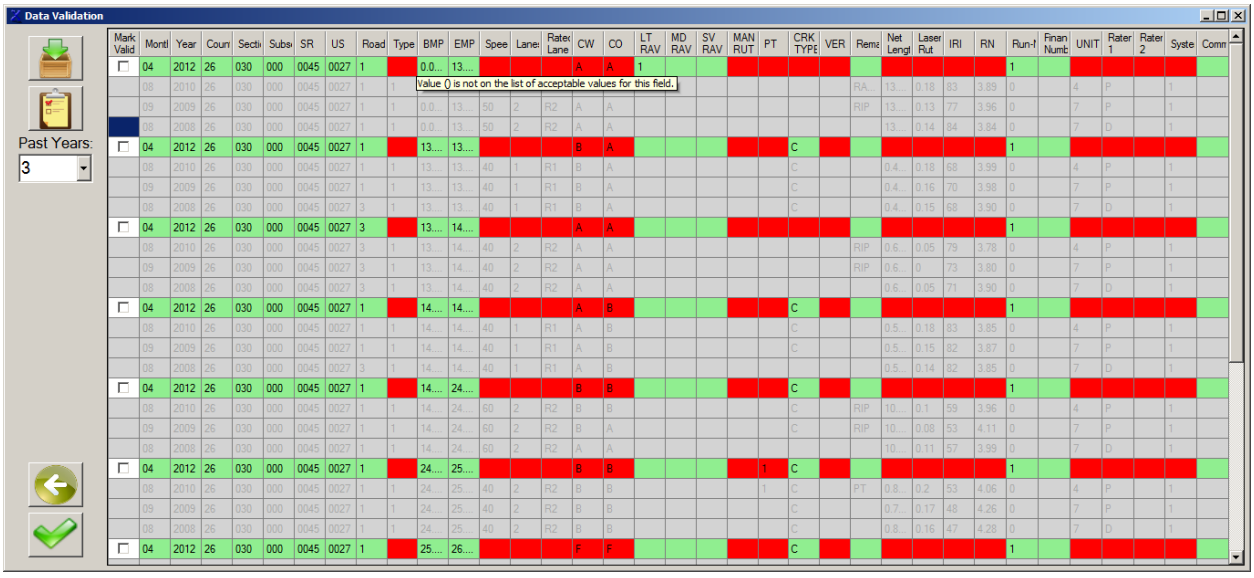

Figure 10. XPCS Mobile Data Validation Screen

Support & Improvement

Per the requirements of the RFP, ARA will provide support of the existing product for a one year period starting at the conclusion of the project. In addition to the support of the existing software, further steps could be taken to improve the overall system.

Improved Enterprise GIS Integration

The XPCS system was developed in parallel with the FDOT Enterprise GIS. While the current system makes extensive use of FDOT GIS data, better integration with the Enterprise GIS system is possible. ARA and FDOT have discussed these needs, and the Department has decided to perform this work within the existing Enterprise GIS integration effort. ARA will continue to assist FDOT with this work through the provision of software and documentation to meet the Department's needs.

Pavement Data Integration

Along with network-level PCS data, there are many other data sources that contain information about a given length of road (i.e., test section). A significant improvement to the system would be providing the ability to display these other data along with the network-level data collected through the XPCS system. Currently, ARA is working with the Department to determine how best to address this need.

Appendix A

Software Requirements Document: Office Component

Table of Contents

List of Figures

List of Tables

Introduction

This document will describe the office component of the Extended Pavement Condition Survey (XPCS) application and requirements which must be met to ensure successful implementation of the component. The office XPCS application is a part of a larger system involving the Roadway Characteristics Information (RCI) database and XPCS survey database which needs to be collected and evaluated at several key steps throughout the data flow process. The XPCS database also includes the subset dataset for Highway Performance Monitoring System (HPMS).

Purpose

The purpose of the office XPCS application is to provide office-based users (supervisors, engineers, management, operators, etc.) with a way to access, map and report the data collected by the PCS survey process. This includes such items as:

- Tabular results, either in a permanent format (i.e., PDF) or editable format (i.e., Microsoft Excel)
- Electronic maps of current conditions (GIS layers)
- Data regarding completion status of the annual survey (percent complete, counties completed, specific segments completed, etc.)
- Standard and ad hoc reporting and mapping functionality.

Furthermore, the office component will provide operators the ability to take data from their vehicle and upload it to the central database through the use of USB drives. It should also provide access to quality control checks for the PCS supervisor(s) that may be run in addition to the checks performed in the mobile component.

Finally, the office component needs to provide supervisors and operators the ability to generate optimized routing plans that can be used in a week's worth of surveys. For the most flexible optimization option, ARA will assist FDOT in learning the applicable parts of ArcGIS Network Analyst. Using this feature of ArcGIS directly will allow the users to perform all the functions listed in the original Request for Proposals.

The office component also includes an upgraded central database. As a part of this effort, ARA will convert the existing PCS mainframe database to a relational database using Microsoft SQL Server 2008. This has the advantage of adding more logic to the database schema (versus a flat file approach). For example, retrieving all the historical test results for a particular Test Section will be far easier now and those results can be ordered in any user-defined manner (oldest to newest, worst to best, only those that include results for the segment between MP 22.8 and 30.2, etc.). This will not affect the original PCS database which resides on the mainframe.

Scope

There are several items that will be provided as part of the office component of the XPCS software.

Each item is discussed in detail within this section.

Database

For these reasons and to improve the general functionality of the overall XPCS software package (both mobile and office components), a copy of the data currently residing in flat tables in the mainframe SAS database will need to be migrated to a new Microsoft SQL Server relational database. This new database will be referred to as the "XPCS Database" throughout the remainder of this document.

Because the mainframe database stores the definition of the State's Test Sections and those definitions are not maintained by the SMO, the XPCS Database will require a link to the existing mainframe database to retrieve these definitions. The office component will contain functionality that displays inconsistencies between the mainframe definition and the XPCS definitions which can then be addressed by the SMO. In most cases, resolving these inconsistencies will require changing the Test Section definitions within the XPCS database. ARA will provide training to the SMO on this process.

Routing

The office XPCS application will create a user customizable optimized routing plan utilizing ArcGIS Network Analyst. The optimized routing order will then be included in the PCS database and used by the mobile component to navigate the network test sections. At this point, the actual routing between various Test Sections will be performed by the mobile component. Sections requiring reruns will be highlighted in yellow in MapPoint and will be marked as incomplete in the database. It will be up to the van operator to decide when to rerun test sections, as the optimized routing plan is developed before the van crews begin their surveying.

Synchronization

Because ARA expects synchronization between vehicle computers and the XPCS Database to be a manual process (download to a USB key then plug into a networked FDOT computer), synchronization functions will be included in the office component of the software.

Validation

Additional quality control checks above and beyond what are done in the mobile component will be performed through the use of the office component of the XPCS.

Reporting

Various electronic maps and reports will be created for the users of the office component of the system. The reports generated as a part of this effort are detailed in sections discussing the dashboard and reporting interfaces. FDOT staff will have the capability to expand the mapping and reporting capability as needed.

Background

FDOT maintains a large RCI database which is used as the "central repository" for managing all information related to roadway characteristics. The profilers collect data throughout the state of Florida and its operators and managers are responsible for the validation and uploading of their collected data

to the mainframe.

This process is currently conducted using Microsoft Excel workbooks and macros. System users "check out" their respectful county data from a master workbook which is itself generated from the PCS database, and transfer the county workbook to the van. The workbook data is then printed out each day and used to verify section and segment data during the profiling process. Data collected by the user on the printed workbook sheets is then manually transcribed after the day's collection ends.

After the collection period for a county is over, users then bring the workbook back into the office and validate the collected data again before utilizing Excel macros for processing and uploading to the mainframe PCS database.

It is the desire of FDOT to automate several of the above processes in their current system, and to provide managers with up to date reports and a "dashboard" style interface for quick review of current network testing status. The office XPCS component implementation will require the PCS mainframe data to be converted to SQL Server 2008, and will provide automated route planning, data validation, and reporting. These features will greatly reduce the complexity of the system and reduce the effort required for managing collected data.

Assumptions

Assumptions for the office XPCS system are:

1) FDOT personnel have access to the RCI database.
2) FDOT personnel wish to migrate their existing PCS database from SAS to SQL Server 2008. The SAS database will continue to be updated and used for other reports, until such time as the SMO feels ready to migrate all reports to the new database structure.
3) FDOT will allow the "mobile computers" containing the mobile XPCS application and synchronized PCS database access to the FDOT network for synchronization purposes, or will manually transfer backups of a SQL Server 2008 database to/from profiling vehicles.
4) ARA will provide, within reason and budget, the necessary hardware/software onboard the profiler for harboring a SQL Server 2008 Express database instance, .NET 4.0 framework, PCS SQL Server database, and the mobile XPCS application (see Constraints, below).
5) ARA will create a database instance specific to the XPCS software.

Constraints

Due to the size and complexity of Network Analyst, ARA will be able to provide only a "nearest next test section" optimization process and provide basic training in its creation. There will be some processes the SMO will need to go through to verify the basemap used to generate optimal routes is correct, and possibly manually assign section visits based on specific conditions (i.e., construction, traffic, etc.). Future work can include expanding the route optimization to include other variables. Routing between test sections will still be performed in the mobile component through MapPoint.

IT concerns may limit what software can be installed on the profiler computers in the event that they

are to be allowed access to the FDOT network. This may require FDOT to purchase separate computers or other hardware for the purposes of this project. ARA will cover, within reason and budget, hardware for the prototype vehicle if needed.

During the course of interviews with FDOT, several issues were discussed related to IT and mobile computers. It was determined that the desire of FDOT was to install the mobile XPCS software on existing machines if at all possible. This will allow FDOT profiler crews to keep the existing setup within the vans. However, this approach also implies that *direct synchronization from the vans* will not be possible unless the computers in the van are brought into the IT definition of mobile devices. This means van computers would need to have appropriate anti-virus software and encryption software installed. Furthermore, the vans would also require the necessary hardware for wireless access. The software listed above may, or has in the past, conflicted with the ICC software/hardware; as such this solution is less attractive. FDOT mentioned it would prefer to have the personnel in the van create a backup of the PCS database, restore that backup *in the office*, then proceed with synchronization.

Methodology

Part of the office XPCS implementation process will be the training of FDOT personnel in the use of ArcGIS Network Analyst. FDOT personnel will use this software to generate an optimized test section list, then use an ArcGIS XPCS plug-in to automatically move the routing data to the office XPCS working directory. As a startup process the office software will prompt the user and, upon authorization, load these data into the XPCS database specified during login. These data will then be synchronized with the van database and will be utilized by MapPoint in the mobile component.

PCS data taken from the profiling vans will be validated at the office and checked against the RCI database. Once a section has passed all validation and verification checks, it is considered “completed”.

Context

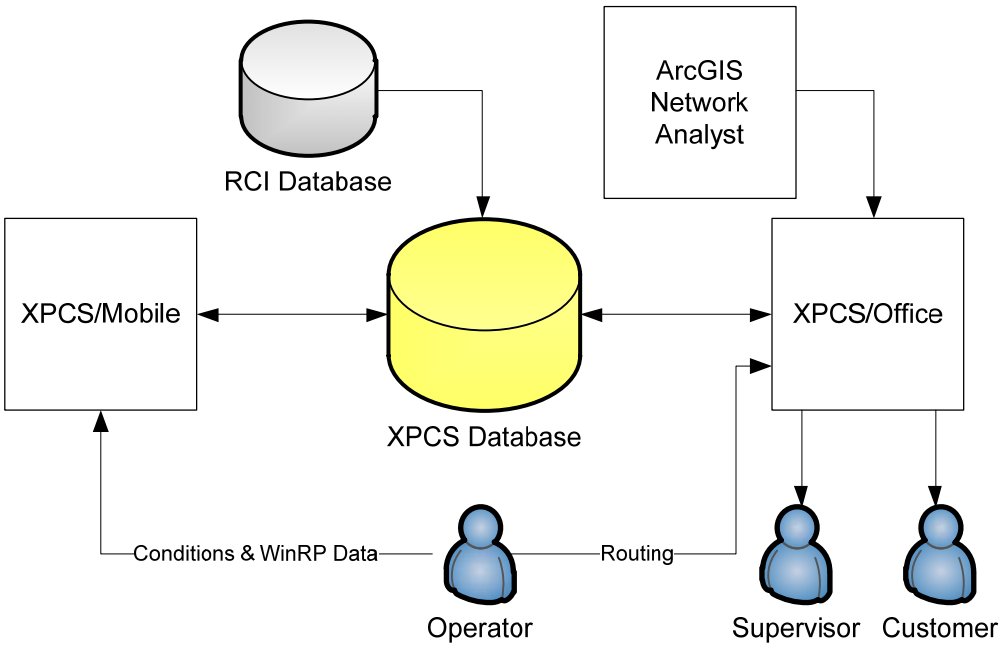

Figure A1. Office XPCS context diagram

As Figure A1 illustrates, the office component is comprised of software whose goal is to fulfill the functions listed in previously under Routing. ARA has selected this extension approach to optimize flexibility and still allow easy manipulation and viewing of the spatial data. All interactions with the XPCS database will occur through the office component.

Office XPCS Data Flow Diagrams

This section will detail the flow of data throughout the XPCS system and will breakdown the context diagram into smaller functional pieces.

RCI/PCS Data Flow

The data flow diagram shown in Figure A2 represents the data validation process between the RCI and XPCS databases. The application will need to access the XPCS Database and validate each section's jurisdiction and mileposts with the data contained in the RCI database. If differences (in jurisdiction or mile posting) exist between the RCI mainframe and the PCS database, the changes from the RCI database are to be reported to the user for modification through a Test Section Consistency report. This interface is described in the section titled Reporting and contains information on automatically applying changes from the RCI database to the XPCS database. Note this update includes managing changes to the section and segment geometries which comprise the GIS data in the XPCS database. These changes will be made by comparing sections/segments with altered endpoints to their respective section/segment shapefiles.

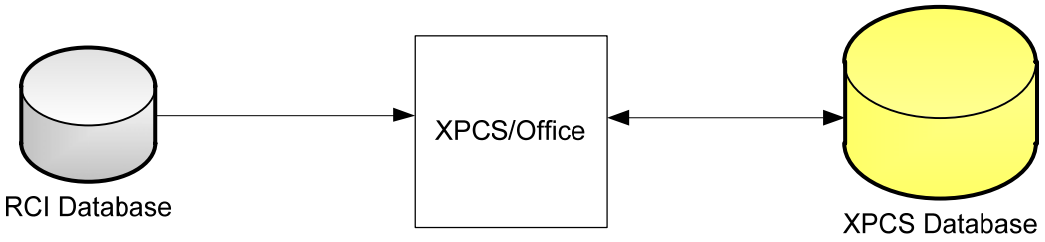

Figure A2. RCI/PCS data validation process

Routing Data Flow

In the process shown in Figure A3, users will need to access or create the FDOT RCI shapefile and pull the data into ArcGIS. This data will be utilized along with data from the XPCS Database to generate an optimal testing order for each Test Section via ArcGIS Network Analyst.

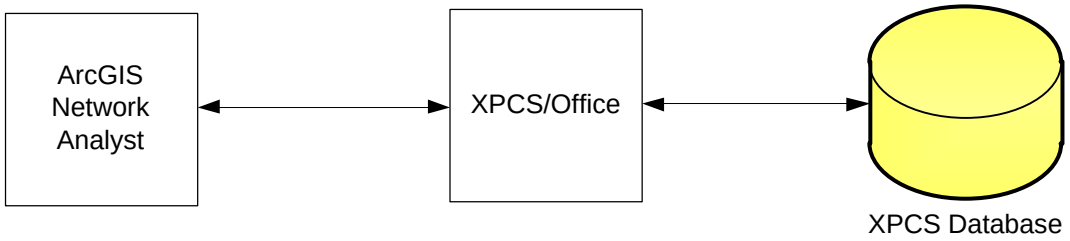

Figure A3. Loading optimized route locations

ArcGIS Network Analyst

A key part of the process we propose in this report involves learning to use ArcGIS Network Analyst to determine the optimum order of test sections in a particular county. Because this may seem daunting at first, we describe the process here in a general fashion to show what is involved. A diagram of the complete process is shown in Figure A4. We have no doubt that learning to use this software will be possible for both supervisors and operators.

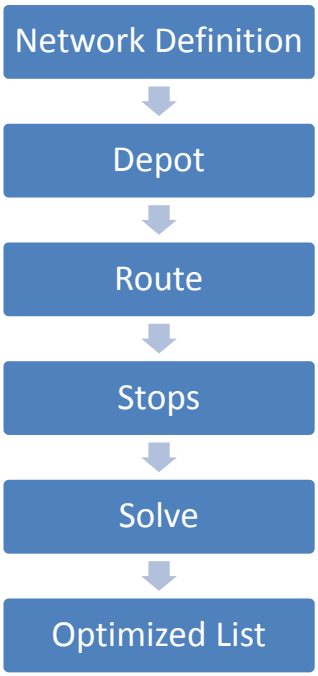

Figure A4. ArcGIS Network Analyst to PCS data flow

The "Network Definition" is the base map that defines the roads the surveyor can use to get to their location. Think of this as the roadmap. Ideally, FDOT already has created this data for other non-PCS applications. If this is not the case, the data available in the Enterprise GIS should be enough to create the network definition. At the worst, there are commercial vendors that can provide this data (e.g., Navteq). Roads are always being added (and sometimes removed from) the network of FDOT and other public roadways, therefore, these geometries will need to be updated on a regular basis via the RCI update interface.

The "Depot", in Network Analyst terms, is the starting point for all vehicles doing work. This should be the hotel or FDOT facility where the operators start each day for a given county. Since operators do not know where they will be starting their work from each day, the optimized routing plan will be based on an initial starting location. The profiling crew can then reorder their test sections on the fly, if it is reasonable to do so, to get back on the optimized testing plan.

"Route", in Network Analyst, represents both the survey vehicle and the route it takes between various points in the network. Even with one vehicle, multiple routes may be needed. For example, if a surveyor is working in Alachua County, they may have four routes (one for each day of survey). This would eliminate the need to mobilize to the last section in the previous day's route. However, it may be easier or more effective to put all test sections on one route in other cases. This will be a matter of testing the different work flows in the field and finding out what works best. When multiple survey vehicles are used (Dade County), multiple routes should be used.

A "stop" represents the beginning or end of a test section. These are the points the survey vehicle

needs to travel through to complete its test. In this case, stops will be paired (*starts* matched with *ends*) to ensure that the route assumes that the vehicle is at the end of the test section when the test is complete.

With the depot, route(s), and stops defined, Network Analyst can solve for the best route. Occasionally, the best route between a start and end stop may not follow the Test Section exactly. This is expected behavior and is not relevant to this implementation.

The solved route(s) (paths through each start and end point) create an optimized list of Test Sections for the surveyor to drive. This list of test sections (and not the route line itself) is what is loaded into the XPCS Database. The MapPoint tool in the mobile component will determine the best route to travel to get to the ordered test sections based on its map at drive time, and will allow for deviations and surveyor rerouting on the fly. Using the actual routes (including directions between test sections) from Network Analyst may become a desired feature for future versions depending on the results of using MapPoint in the field.

Data Conversion Requirements

As a part of the office component development, ARA will work with FDOT personnel to create the initial XPCS Database. This process will involve copying the data from existing flat SAS datasets to a relational SQL Server 2008 database. Eventually, all pertinent PCS reports will be converted from SAS to SQL Server scripts. Not all of the SAS scripts will be converted during this first project. This means "legacy" SAS datasets and reports will still be required (initially). Because of this, a process for pulling the data from XPCS Database into SAS datasets will need to be in place. FDOT will be responsible for the SAS dataset import from the XPCS database. ARA will work with FDOT to determine the best method for exporting the data from SQL Server to SAS datasets. Further discussion on migrating existing SAS reports to the new system are described in the section titled Reporting Requirements.

Logical Data Model

The logical model for the XPCS database, which will store all data not contained in the FDOT Enterprise GIS for the office component of the XPCS software, will be defined as part of the Design Phase of this project.

Functional Requirements

The functional requirements are discussed in Table A1 and will describe a specific use case or user system interaction and derive a verbal requirement which describes the various features and priorities of the office XPCS system.

- **C: Critical.** The feature must be included to ensure the functionality of the core capabilities of the software.
- **D: Desirable.** The feature would improve accuracy of data, time to collect, and/or ease of use.

The feature does not significantly impact the core capabilities of the software.

- **F: Future.** This feature would change the scope of work but may be useful in future versions of the software.

Table A1. Functional requirements

#	Description	Priority
1	GIS plug-in must export test section locations into a file for the Office XPCS application to load into the selected XPCS database in order of their sequence number produced during the optimization process.	C
2	Validate section/segment limits against the RCI mainframe.	C
3	Validate jurisdiction for each section against the RCI mainframe.	C
4	Store section spatial/geometric data.	D
5	Store section begin/end locations as GPS data.	C
6	Allow simple (closest section) optimization of test sections.	C
7	Allow multiple attribute optimization of test sections.	F
8	Allow users to modify all collected data.	C
9	Allow users to modify segment limits.	C
10	Automation of all ArcGIS processes for generating optimal routes.	F
11	“Dashboard” view of % sections completed.	C
12	“Dashboard” view of % sections on time.	C
13	“Dashboard” view of % sections validated.	C
14	Provide GIS layer showing testing progress	C
15	GIS map allows users to select sections to generate section status reports.	D
16	Must allow users to generate section validation report against the RCI database	C
17	Must allow users to generate data validation report against historical (previous years) data.	C
18	Must allow users to generate FPCS/RPCS report from the PCS database.	C
19	Must allow administrators to be able to define data validation thresholds.	C
20	Data validation thresholds must be pushed out to mobile XPCS databases	C

Interface Requirements

Interface requirements describe "cases of use" for each interface in the office XPCS application. Several use cases and screen shots are provided below to better highlight system processes, and to more specifically define their requirements.

Software Interfaces

Figure A5 illustrates the main interfaces in the office XPCS system. The ArcGIS plug-in can exist as a part of the ArcMap interface or in the office XPCS application. Please note that there is a one time extensive data conversion process for migrating the PCS data from its current database to the SQL Server 2008

database. This process has no user interface and is not displayed below.

Figure A5. Office XPCS components

Login Interface

First, the users will be presented with the login screen where they can identify themselves and perform general setup operations. The last user to log into the system can login just by clicking the "Login as…" button. They will then be prompted to enter their password. If this is the first time the software is used on a particular machine, this button will be disabled and read "Use Setup to login". Users will be able to switch user names from the "Setup" button. Clicking the "Setup" button also allows users to change system parameters including, database login and connection information, XPCS usernames and passwords, and other system wide user parameters. Exit quits the program. Table A2 describes each of the functional requirements of the interface while Figure A6 displays a mockup of a suggested login interface.

Table A2. Login interface requirements

#	Description	Priority
Login Interface		
1	Maintain user name across sessions.	C
2	Maintain database connection information across sessions.	C
3	Allow users to set up database connections to mobile XPCS databases.	C
4	Provide users with information rich feedback on login/setup issues.	C

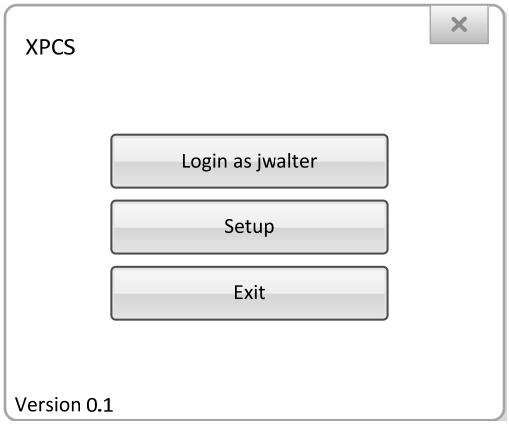

Figure A6. User "jwalter" logs in to the Office XPCS application

Office XPCS Navigation Interface

This interface, shown as a mockup in Figure A7, allows the user to select which part of the office XPCS application they wish to navigate to. The interface should allow the user to navigate to any other part of the application including the login interface.

Figure A7. Office XPCS navigator

Security/User Management Interface

This interface allows system administrators to set up users for the XPCS system. System security will consist of two classes of users, administrators and users. Administrators will have access to the office XPCS interface and all of its features, users will have access only to the synchronization interface. In the diagram below, we see the user "jwalter" is a member of the administrators group, but not a part of the users group.

In the mockup shown in Figure A8, administrators will be able to click the "Create New User…" button and enter a new user name and password in the PCS database. The user should be asked to change their password the first time they log in. Administrators should also be able to delete an existing user by pressing the "Delete User" button.

Pressing "Finish" in the interface commits any changes made to the PCS database. Pressing "Cancel" cancels any changes made in the interface. Note that since Administrators and Users have predefined actions in the system, there is no need for managing further user access capabilities. This interface and resulting database structure can be modified if future changes to user permission features are required.

Table A3 discusses the functional requirements related to the security features of the program and the interface used to control those features.

Table A3. Security requirements

#	Description	Priority
Security/User Management Interface		
1	Manages two types of account access: users and administrators.	C
2	Application must allow for the creation of new accounts.	C
3	Application must allow for the deletion of accounts.	C
4	Application will allow for cancelling changes made.	C
5	Administrators will have access to all features and functions in the application.	C
6	Users will be able to modify, validate, and synchronize PCS data only.	C

Figure A8. Security/User management interface

Dashboard Interface

The dashboard interface will show an instant report of sections tested, sections validated, and sections completed vs. scheduled time. The dashboard can be updated on demand. Updates pull data from the XPCS database as it currently exists. Table A4 lists the requirements for this interface while Figure A9 shows a mockup of the suggested dashboard.

Table A4. Dashboard requirements

#	Description	Priority
Dashboard Interface		
1	Shows instant report of sections tested.	C
2	Shows instant report of sections validated.	C
3	Shows instant report of sections completed on time.	C
4	Report can be refreshed on demand.	C
5	Data is updated from the PCS database.	C
6	Can be easily viewed/hidden.	C

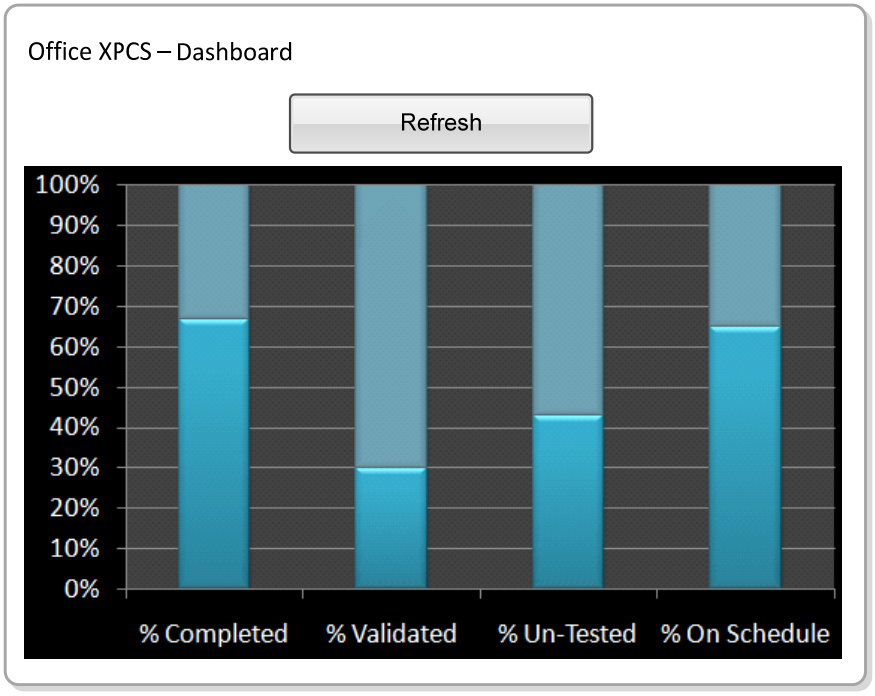

Figure A9. Dashboard interface

Reporting Interface

The reporting interface maintains selections between various reports created by ARA and FDOT for the XPCS application. ARA anticipates FDOT creating several reports after major development of the XPCS system is completed. Functional requirements for reporting are shown in Table A5 and a mockup of the interface is shown in Figure A10.

Table A5. Reporting requirements

#	Description	Priority
Reporting Interface		
1	Allow users to select from various reports.	C
2	New reports can be added to the Reports Management interface.	F
3	Generating a report runs custom report code to create the currently selected reports.	C
4	Interface must allow users to cancel report generation and return to main screen.	C
5	FPCS report converted from SAS to SQL Server 2008	C
6	RPCS report converted from SAS to SQL Server 2008	C
7	Lane Miles surveyed on the state maintained roadway system report converted from SAS to SQL Server 2008 (two types)	C
8	RCI Verification report converted from SAS to SQL Server 2008	C

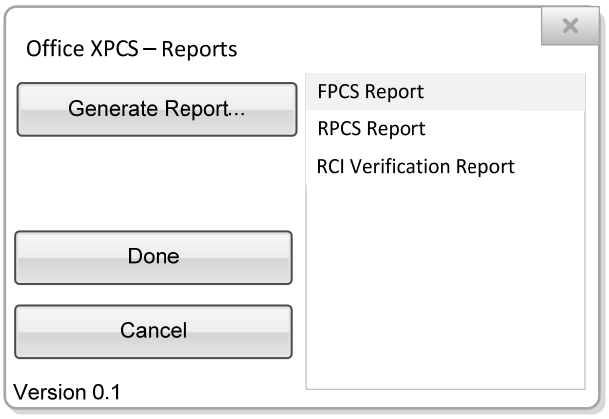

Figure A10. Report management interface

Flexible PCS (FPCS) and Rigid PCS (RPCS) Reports

The FPCS and RPCS reports are general reports detailing each segment and its associated rating data as given in the FPCS.xlsx and RPCS.xlsx spreadsheets and associated SAS documentation. ARA will convert the SAS code to SQL Server queries and generate reports with fields and data matching those of the required SAS reports.

RCI/PCS Validation Report

This report requires users to connect to the RCI database and to validate each PCS section's jurisdiction and mileposts (realignments). This process generates a report showing changes made to the PCS database from the RCI mainframe. ***Note that this report is also generated in the Data Management Interface, when a user selects "Verify and Load RCI Changes".*** Choosing to run this report from the report interface does not alter any PCS data whereas running the "Verify and Load RCI Changes" will modify the XPCS database. If modifications are required, the user will be propmpted to load shapefiles containing the (possibly) new section and segment geometries.

Data Management Interface

The data management interface allows users and administrators to view and modify the existing PCS data in the system in a tabular format. The interface should allow users to view "queried" or "filtered" data. The interface should allow users to run a validation check against selected data. This interface should also highlight, in accordance with FDOT standards (as described in the Field Handbook and FlexWorkbookPointOfEntryChecks.docx), fields which do not pass validation checks. The interface should allow users to view all historical data for a selected section. Part of this interface should look similar to the Flexible and Rigid workbooks used by the profiling crews.

The RCI validation and consequent update to section limits will require the application to modify the existing section and segment geometries stored within the database. As part of the RCI update process, the user must load a SECTIONS.shp and SEGMENTS.shp shapefiles, which represent the changes made in the RCI database. Once selected, these files will update the geometries in the section and segment

tables to match their new RCI beginning and end milepost locations.

A summary of the functional requirements is shown in Table A6 and an interface mockup is provided in Figure A11.

Table A6. Data management requirements

#	Description	Priority
Data Management Interface		
1	View PCS data in format similar to existing worksheets in accordance with FDOT standards. (Highlighting rules given in FlexWorkbookPointofEntryChecks.doc.)	C
2	Modify PCS data.	C
3	Validate selected rows of PCS data.	C
4	Validate all PCS data.	C
5	Verify PCS jurisdiction and mileposts with RCI database.	C
6	Verify selected rows with RCI database.	C
7	Allow filters to be created for PCS header data.	C
8	Allow filters to be saved.	C
9	Allow filters to be loaded.	C
10	Only allow changes to go through if all data is properly validated, verified, and/or commented.	C
11	All changes made to be rolled back until “Submit Changes” is clicked.	C
12	Allow users to see data for both flexible and rigid pavements.	C
13	Allow users to view historical data for selected sections.	C
14	Allow users to update sections database table from a shapefile provided by FDOT.	C
15	Allow users to update segments database table from a shapefile provided by FDOT.	C

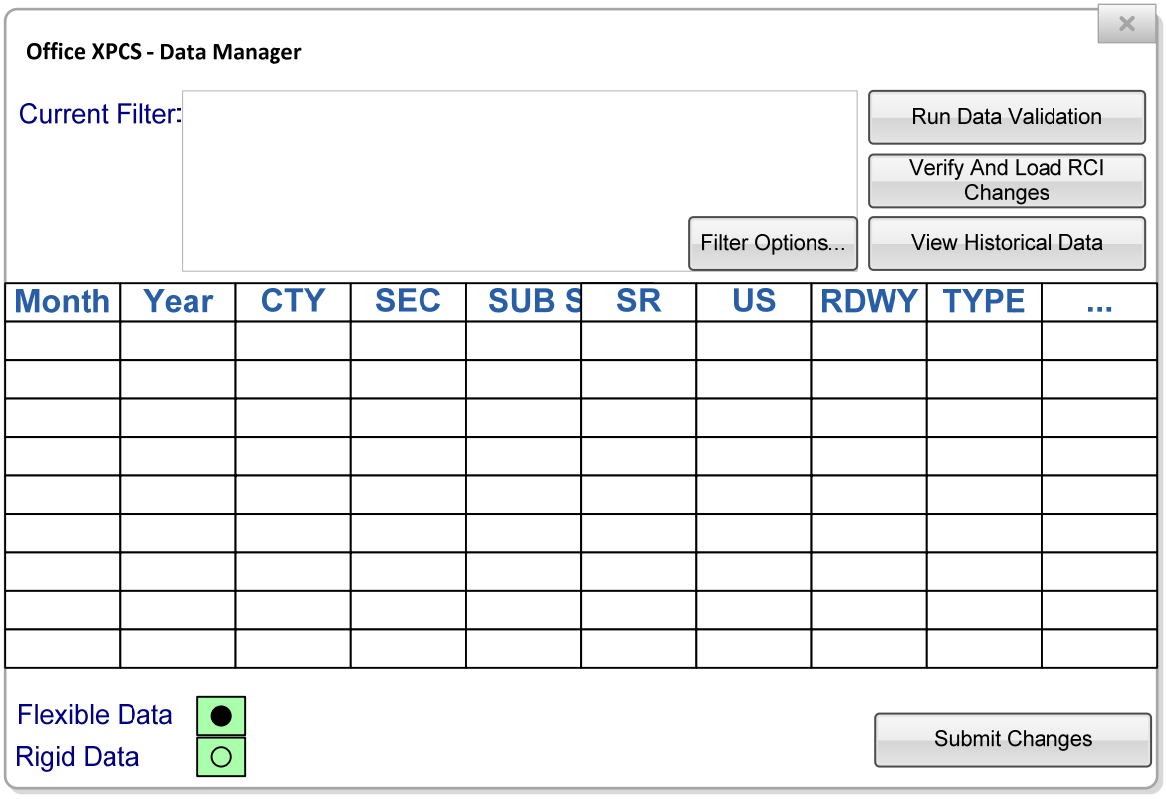

Figure A11. Data manager interface

ArcGIS Plug-In Interface

The ArcGIS plug-in's primary purpose is to allow the user to seamlessly transfer data between a shapefile containing optimized section information to the XPCS database. The ArcGIS plug-in will launch from within ArcGIS. Functional requirements for this interface are shown in Table A7. The interface for this functionality will be very simple: a button inside of ArcMap (as shown in Figure A12) will transfer data between ArcGIS and the XPCS system.

Table A7. ArcGIS plug-in requirements

#	Description	Priority
ArcGIS Plug-in Interface		
1	Must allow user to export optimized section list to the XPCS data folder.	C
2	Must allow user to load test section precedence into the XPCS database.	C

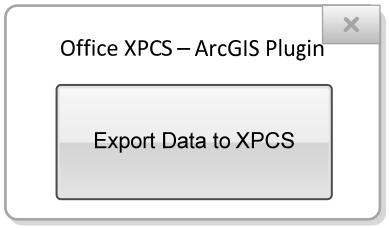

Figure A12. ArcGIS plug-in interface

Test Section GIS Interface

The desktop software will utilize SharpMap to reference FDOT street data, display test sections and link directly to the XPCS database. The interface will allow users to manage multiple layers of data and export the PCS attribute data including unique roadway ID's to a shapefile. This shapefile can then be used in ArcGIS to link with other data sets. Functional requirements for this interface are shown in Table A8.

Table A8. Test section GIS requirements

#	Description	Priority
Test Section GIS Interface		
1	Show all test sections in the state as line geometries.	C
2	Highlight test sections according to processing level. (Incomplete – Yellow, Tested – Blue, Validated – Green, Not Tested – Red)	C
3	View section details from the GIS map.	D
4	Manage multiple layers in the GIS map.	C
5	Filter for desired sections based on county, section, subsection and mileposts.	D
6	Export selected layer as shapefile	C

Data Synchronization Interface

This interface should allow users to synchronize the office XPCS database with one or more mobile XPCS databases. Synchronization should only be possible if all data has been properly verified against the RCI mainframe and validated against previous years data. Functional requirements for the data synchronization process are listed in Table A9. Figure A13 shows an example synchronization interface as used by a similar ARA program.

Table A9. Data synchronization requirements

#	Description	Priority
Database Synchronizer		
1	Interface must allow users to input local server names.	C
2	Interface must allow users to input local database instance names for batch synchronization runs, or single local database instance names for a single synchronization run.	C
3	Interface must allow input of central server name (the “to computer”).	C
4	Interface must allow input of central server instance name (the “to computer” database name).	C
5	The synchronizer must synchronize data between databases using a last modified data flag for updatable data.	C

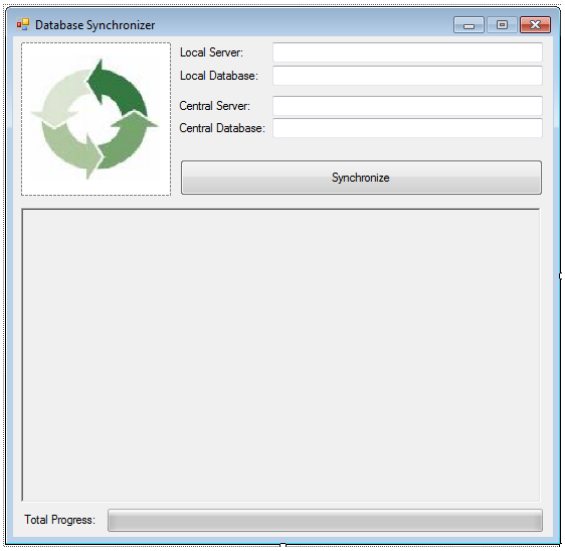

Figure A13. Office XPCS database synchronizer

Hardware Interfaces

1) Laptop or Desktop computer running Windows XP SP2 or better.
2) Server grade computer for holding the central XPCS database.

Communication Interfaces

1) Ethernet or wireless connection to the server hosting the XPCS database.
2) XPCS Data Synchronization routine.

Non-Functional Requirements

Non-Functional requirements (NFRs) of a system typically describe qualities of the system. Requirements such as, performance, security, data management, etc. are all non-functional

requirements. Each of the pertinent non-functional requirement headings below discuss the office XPCS NFRs.

Hardware/Software Requirements

The office XPCS database will be hosted on a virtual server at the SMO. Access to the XPCS database will be managed through the SMO's IT staff. The XPCS system requires one or more machines capable of running Microsoft SQL Server 2008 and ArcGIS with the Network Analyst extension.

IT Requirements

No known special IT requirements.

Security/Privacy Requirements

Security for the office XPCS system is managed at two levels, the application itself, and the database. Users and administrators will need access permissions to both in order to use the system. Users may only perform synchronization and data editing functions in the office XPCS application. Administrators shall be granted full access to all system features.

Data Capacity

The SQL Server database holding the PCS data must not exceed 10 gigabytes in size, or it will cease to function with the mobile XPCS SQL server express databases. With this design and reasonable database maintenance (to be carried out by SMO IT staff) the database should remain less than 1 GB in size.

Data Retention

Data should be retained in the database until archived by FDOT personnel.

Data Conversion Requirements

The data load from the existing SAS datasets and Excel worksheets is a one-time data load. It is the desire of FDOT to continue the data conversion process from SAS to SQL Server 2008 in future projects.

Error Handling

The application must provide information rich errors back to the user interface, and perform all synchronizations and other data transfers with "rollback" capabilities.

Validation Rules

All data validation shall be performed in accordance with FDOT standards as described in the PCS Handbook, FlexWorkbookPointofEntryChecks.doc and RigidWorkbookPointofEntryChecks.doc documents.

Appendix B

Software Requirements Document: Mobile Component

Table of Contents

List of Figures

List of Tables

Introduction

This document will describe the mobile component of the Extended PCS (XPCS) software and requirements which must be met to ensure its successful implementation. This component is a part of the overall XPCS system which consists of a central database containing all pertinent data and an office component for routing, reporting, quality control, and tracking.

Purpose

The mobile component will be used onboard FDOT ICC data collection vehicles to identify test sections, route the vehicle to those test sections, and collect condition data from both the operator and WinRP. Additionally, the system should provide tools for logging, validation, visual displays, and vocalization of route directions. Finally, it shall synchronize an onboard database with the data contained in the central XPCS database.

Because the data collection vehicles and their crew are expected to operate independent of any connection between the FDOT IT network and the onboard computers, a synchronization process will be required. The synchronization process will transfer newly collected data to the central XPCS database and update the vehicle database with the current version of the central XPCS data store. The data collection vehicle may carry a portion of the data in the XPCS database or a complete copy of the XPCS database. This synchronization process will utilize USB keys to transfer the mobile XPCS database to the SMO. The office XPCS application will have a synchronization interface to perform the actual synchronization. As a part of possible future work, FDOT may wish to enhance this process by allowing van crews to synchronize data directly from the profiling van, (and directly from the mobile XPCS application), at the SMO via the wireless network.

Scope

The mobile component will provide the following core functionality:

- Allow the Office XPCS application to synchronize with the Mobile XPCS database.
- Provide routing directions to the next test section based on a routing plan developed by the office component.
- Allow the operator to override routing for unexpected issues.
- Read WinRP Data into the mobile XPCS database and various user interfaces.
- Update the distress data with input from the vehicle operator.
- Collect commentary on the section through both standardized comments and free-form commentary.
- Implement the Mobile XPCS system on one (1) survey vehicle, keeping in mind expanding the project to other vehicles.

Background

This section will describe the project background and any existing processes currently in practice at the

SMO.

Existing Processes

Currently, pavement condition surveys (PCS; which also refers to the current software system used at the SMO) are performed by a single person crew in an ICC profiler. Five vehicles and their respective operators work from the SMO. The surveyor selects a county and gathers all required data for a single work week. While most counties have a single surveyor assigned to them, a few use multiple surveyors for high density areas (e.g., Miami-Dade County). No cross-county routing required.

Electronically, the surveyor has a Microsoft Excel spreadsheet describing the county they are working in and the ICC profiling setup, which primarily creates text files that can be imported into the electronic Excel file and checked according to specific QC processes. On paper, the surveyor carries a copy of the Excel sheet for notes and surface condition observations. These observations, along with any changes to the test section (new pavements, construction areas, segment breaks, etc.), are entered into the electronic version of the spreadsheet after the survey.

The surveyor also carries a map of the county and a printout from the web based GIS system of each test section in the county. Exactly what the surveyor carries differs by crew person; some latitude has been granted to the crews to determine how to get to their sections and survey them effectively.

Network Definition

FDOT tests the network in accordance with the test sections defined in the Roadway Condition Information (RCI) database. Each test section is unique to a county and usually follows a single posted route number (e.g., State Route 114). When signed routes cross county lines, the current test section ends and a new one begins. Each test section is also linearly referenced in units of miles with a start typically at 0. Test section identification numbers are usually split into three items:

- A two-digit county code
- A three-digit section ID
- A three-digit subsection ID

As an example, section 26005000 is test section 5 in Alachua County (County ID 26). Section 26005001 is a test section that is a spur or realignment from test section 5. While 26005001 is *typically* geographically coincident with 26005000, in both business processes and data storage, it is considered a separate test section. Therefore, for the purposes of this document **test subsections are considered to be functionally equivalent to test sections.** In data terms, an individual test section is identified by a combination of county ID, section number, and subsection number.

A standard practice in the management of pavements is to work with an area of pavement that is consistent throughout (e.g., same cross section, material, and construction project). For FDOT purposes, each test section needs to be divided according to the following pavement attributes:

- Surface type

- Number of lanes
- Apparent age (surface condition should be uniform throughout the length)
- Known construction projects

As most test sections will have variations in the attributes shown above, the test sections must be divided into segments where these factors remain the same. In the current system, a segment is matched to a single row in the Excel file. The point where one or more pavement attributes changes is called a breakpoint. A breakpoint automatically exists at the following locations in a test section:

- The beginning of the test section (Milepost 0)
- The end of a test section
- At the start of any discontinuity
- At the end of a discontinuity

All segments start and end at a breakpoint. The list of test sections is contained in the RCI database. However, the breakpoints and segments within each test section are maintained and updated through the PCS process.

Assumptions

1) Windows XP Service Pack 2 or later or Windows 7 shall be the operating system utilized by the Extended Pavement Condition Survey (XPCS) application.
2) All users will fall into one or more of the following roles:
    a. Operators
    b. Administrators
3) Any computer running the mobile component will have SQL Server 2008 Express Edition installed.
4) A base map, suitable for use with Network Analyst will be provided by FDOT.
5) Test sections will be identified on the centerline referenced in Item 4.

Constraints

1) The mobile component will operate independent of the FDOT network.
2) WinRP must be operated manually.

Methodology

The functional requirements in this document were produced by interviewing FDOT and ARA van crews, specialized ICC data collection personnel, project supervisors, and FDOT IT staff.

Context

The following is a context diagram of the XPCS software suite and its related external systems.

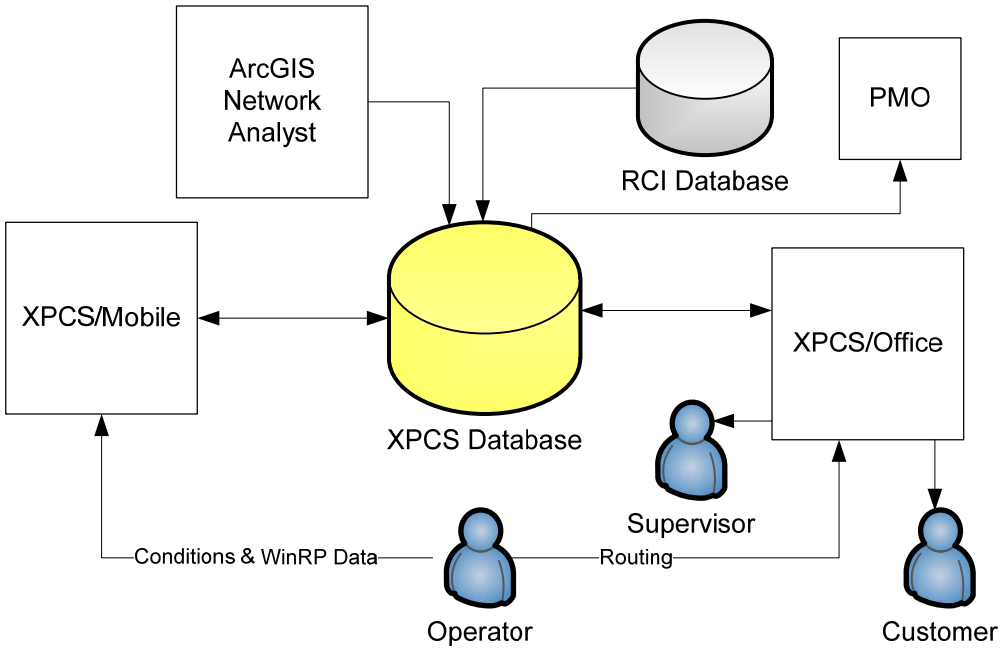

Figure B1. Mobile XPCS context diagram

As can be seen in Figure B1, all components of the system use the central XPCS database. As the survey vehicle operates disconnected from the central database, the mobile component will need to move data from the central database into a full or partial copy of the database on the vehicle. The routing element consists of the entire route plan for the area (usually a county) to be collected. The application should visually track the current test section and current segment during testing as well as the van's current location. ARA plans to use Microsoft MapPoint to direct users to test sections with its existing text to speech capabilities and to display the vehicle's current GPS location.

MapPoint was selected as the onboard navigation utility for several reasons. First, field operators are already familiar with the MapPoint interface making transitioning to the Mobile XPCS system easier. Second, MapPoint features easy to control routing capabilities, an integrated software development kit (SDK) which allows the system to be pre-programmed with routing instructions, a built in TTS (Text to speech) engine, and allows users a much more rich interaction with the software. MapPoint can also be configured to accept and display ESRI-based shapefile data (with some custom add-ins). Finally, MapPoint is *much* less expensive than its ESRI and other competitors counterparts making it less of a financial burden for expanding the system to other FDOT vehicles.

Mobile XPCS Data Flow

Figure B2 describes how data moves through the mobile XPCS system.

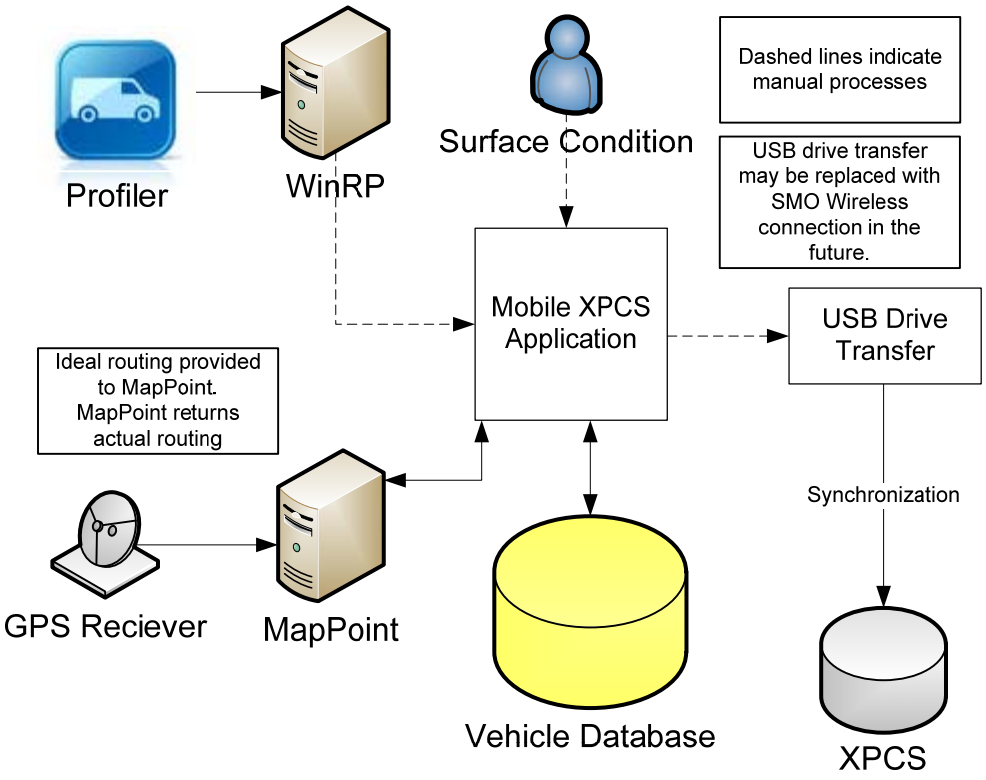

Figure B2. Mobile XPCS data flow diagram

Optimized routing instructions from the office component are used by the test section routing interface to enable the van crew to receive directions to the next test section. Field Crew modifications (for unexpected traffic, construction, etc.) to the optimized routing list are performed through the mobile XPCS interface. Data from the profiler is then provided to the software via WinRP on the ICC computer. The van operator imports this data by identifying the appropriate text file. The operator also inputs surface condition data either during the run or at the end of the run via the software interface. Once data has been processed via WinRP, the software runs automated validation checks against the profile and surface condition data. Van operators are then able to make changes to section and segment data and may also choose to re run a specific section or segment if bad data was generated. Any changes to section or segment data, and reruns performed will be permanently recorded (logged) and cannot be altered.

Logical Data Model

Table B1 describes any tables or transactional data structures used by the mobile component.

Table B1. Data model requirements

#	Description
1	Each test section is identified by a unique combination of county ID, section number, and subsection number.
2	Must provide data storage for section validation checks.
3	Must provide data storage for section quality assurance and quality control checks.
4	All test runs must be saved in the database in their original form. (i.e., Changes to the data during validation or reruns must be stored separately).
5	Must provide data storage for reasons why collected data has failed validation checks.
6	Must provide data storage for all PCS related data.

Functional Requirements

The functional requirements, as shown in Table B2, will describe a specific use case or user system interaction and derive a verbal requirement which describes the various features of the mobile component. Each functional requirement describes a feature in the system and its priority:

- **C: Critical.** The feature must be included to ensure the functionality of the core capabilities of the software as described within the scope described previously.
- **D: Desirable.** The feature would improve accuracy of data, time to collect, and/or ease of use. The feature does not significantly impact the core capabilities of the software.
- **F: Future.** This feature would change the scope of work but may be useful in future versions of the software.

Table B2. Functional requirements

#	Description	Priority
Section Identification		
1	Reduce the time and complexity of identifying the start and end of a test section.	C
2	Pull all pertinent survey information for test sections the surveyor plans to evaluate in a specific trip into the vehicle database.	C
3	Import optimized routing instructions.	C
4	Display optimized routing instructions in MapPoint.	C
5	Display current county.	C
6	When traveling to a test section, display the next test section's ID and starting location. ("Next test section" refers to the next test section to be visited).	C

Table B2. Functional requirements (con't)

#	Description	Priority
7	When testing, display the test section ID and the end location.	C
8	When testing, allow the operator to increase and decrease the segment.	C
9	Suggest current segment based on GPS location.	C
10	Display & edit the State Route Number.	C
11	Display & edit the US Route Number.	C
12	Display & edit roadway testing direction (ascending/descending).	C
13	Display & edit roadway division (composite/divided).	C
14	Display & edit section status.	C
15	Display beginning milepost for current segment.	C
16	Display ending milepost for current segment.	C
17	Add breakpoint at user defined milepost.	C
18	Add breakpoint at current location.	D
19	Display & edit testing speed.	C
20	Display & edit number of lanes.	C
21	Display & edit rated lane.	C
22	Display & edit pavement type.	C
23	Change interface based on type of pavement: rigid or flexible.	C
Flexible Surface Condition Inspection		
24	Input observed wheel path cracking code.	C
25	Input observed non-wheel path cracking code (outside wheel path).	C
26	Input the predominant raveling level (may be none).	C
27	Input the raveling extent (if raveling level > none).	C
28	Input a manual rutting rating (only after completion of run).	C
29	Input patching extent.	C
30	Input area cracking presence (may be none, alligator, block, or a combination).	C
31	Display verification flag.	C
32	Display & edit standard remarks.	C
33	Display net length (length of section minus any length not recorded due to roughness off/on flags).	C
34	Display profiler rutting.	C
35	Display profiler IRI.	C
36	Display profiler Ride Number.	C
37	Display previous year values for items 25-31 and 33-37.	C
38	Input free-form comments.	C
Rigid Surface Condition Inspection		
39	Input number of slabs affected by transverse cracking (light, moderate, and severe).	C
40	Input number of slabs affected by longitudinal cracking (light, moderate, and severe).	C

Table B2. Functional requirements (con't)

#	Description	Priority
41	Input number of slabs affected by spalling (moderate and severe).	C
42	Input number of slabs affected by corner cracking (moderate and severe).	C
43	Input number of slabs affected by patching (fair and poor).	C
44	Input number of shattered slabs (moderate and severe).	C
45	Input number of slabs with surface deterioration (moderate and severe).	C
46	Input number of slabs affected by pumping (light, moderate, and severe).	C
47	Input joint condition.	C
48	Input estimated slab length.	C
49	Input estimated number of slabs with multiple cracks.	C
50	Calculate/ estimate number of slabs (Item 49)/(EMP-BMP).	C
51	Calculate/ estimate the percentage of cracked slabs in a segment (Items (40+41+43+45) minus Item 50 divided by Item 51 as shown in the flexible and rigid spreadsheets and described in the DataDictionary.xlsx document.	C
52	Display profiler IRI.	C
53	Display profiler Ride Number.	C
54	Display previous year values for items 40-54.	C
55	Display verification flag.	C
WinRP Integration		
56	Automatically run WinRP when a test section is completed.	F
57	Automatically pull in WinRP data from WinRP text files to the RCI database.	D
Automated Validation		
58	Validate collected data with the vehicle database.	C
59	Identify segments which need to be retested before leaving test section.	C
60	Commit section test data to the vehicle database.	C
61	Show % test sections completed.	C
62	Show % test sections on schedule.	D
63	Show % test sections validated.	C
Quality Control		
64	Check collected data against base quality standards.	C
65	Migrate data quality standards from XPCS database (should be read only for the mobile software).	C
66	Visually flag erroneous data.	C
67	Synchronize vehicle database with XPCS database.	C
GPS Tracking		
68	Automatically track the data collection van's location on a GIS map.	C
69	Provide GPS data for MapPoint.	C
70	Display the vans location on a map at all times to the user.	C
71	Identify the most efficient route to the desired test location on the GIS map.	C

Table B2. Functional requirements (con't)

#	Description	Priority
72	Issue audible commands via text to speech to guide the data collection personnel to the desired test location.	C
73	Show nearest test section on interface.	C
74	Show next test section on interface.	C
Application Logging		
75	Log user usage time, miles traveled, and miles surveyed.	C
76	Log total miles traveled based on administrative reset.	C
77	Log total miles surveyed based on administrative reset.	C
78	Log system errors.	C
79	Log all synchronization and validation events.	C

Interface Requirements

Interface requirements describe "cases of use" similar to the functional requirements, but do so for each interface in the mobile component. Several use cases and screen shots are provided below to better highlight system processes, and to more specifically define their requirements.

Interface Cycle

The general design of the software will follow a four step process as shown below in Figure B3.

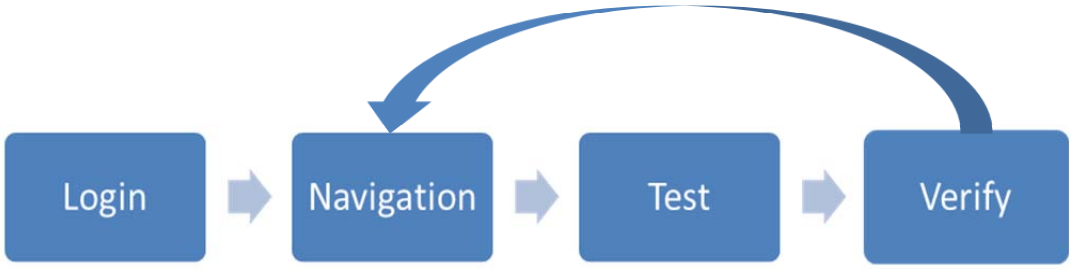

Figure B3. Basic process for software interface

Each phase logically follows the prior phase throughout the testing day. Each element is described below:

- Login – Identify the user, connect to the database, miscellaneous setup options, and any other tasks a user would usually perform outside the testing cycle. These tasks are generally expected to be performed at the start of the work day.
- Navigation – This interface is used when moving between testing sections. It shows how to get to the next optimal test section and allows the user to alter the order of test sections to visit on the fly.
- Test – This interface can be used during testing. It is also designed to be used after testing to

enter specific data (such as surface conditions) on a segment-by-segment basis

- Verify – This interface is designed to be used at the end of a run. Users will need to load the WinRP data from the section run into the XPCS system. The users will then run automated validation checks against the WinRP data.

Software Interfaces

The following interfaces relate directly to actions the user performs when inputting data into the Mobile XPCS system.

Login Interface

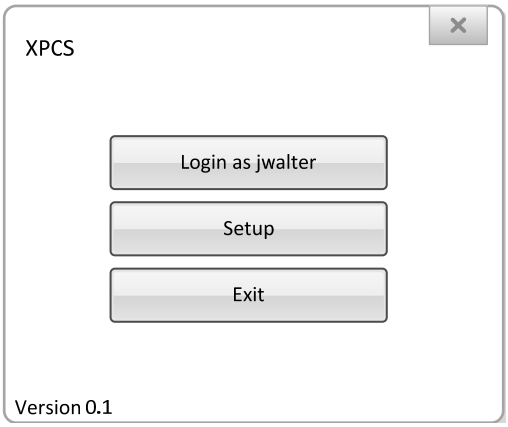

Figure B4. User "jwalter" Logs in to the XPCS Application

First, the user will be presented with the login screen (as shown in Figure B4) where they can identify themselves and perform general setup operations. This screen is meant to minimize the amount of time the user spends in setup. The last user to have logged into the system can login by clicking the "Login as <username>" button. They will then be prompted to enter their password. If this is the first time the software is used on a particular machine, then the "Login as <username>" button operation will automatically take them to the "Setup" interface. From the "Setup" interface, users will be able to switch user accounts, change system parameters, (including database login and connection information and XPCS usernames and passwords), and modify other system wide user parameters. Exit quits the program. Note that user based settings are not synchronized with the central database as those settings need to remain specific for each system user.

Table B3 lists the functional requirements related to the login interface.

Table B3. Login interface requirements

#	Description	Priority
Login Interface		
64	User must be able to change/reset their password.	C
65	User accounts must be maintained in the database.	C
66	Users must be able to change what database they are connecting to.	C
67	User settings should not be synchronized with the central PCS database.	C
68	User names should be saved between sessions.	C
69	Database instance name should be saved between sessions.	C
70	Database login user name should be saved between sessions.	C
71	Login failures should provide users with information as to why the login failed.	C
72	Users must also create a security question related to their password.	C
73	Administrators must be able to reset user passwords.	C

Navigation Interface

In Figure B5 below, we show an example of the interface as the user travels from their current location to the next testing section. At this point, optimal routing information has been provided to the application, and the user simply follows the onboard directions to the next test section. The user can select and view test sections from this interface. Test sections will initially show with red highlighting until completed, yellow until validated, and green once validated and completed. The order in which to visit the test sections appears in the user interface on the left side of the screen. The county button in the center top of the interface can be used to navigate to a different county. It can also be used to select a different route if there are multiple routes in a single county. The “Start Testing” button is used to move to the next phase of the section testing process. The “Setup” button takes the user back to the login screen. Table B4 lists the requirements associated with this interface.

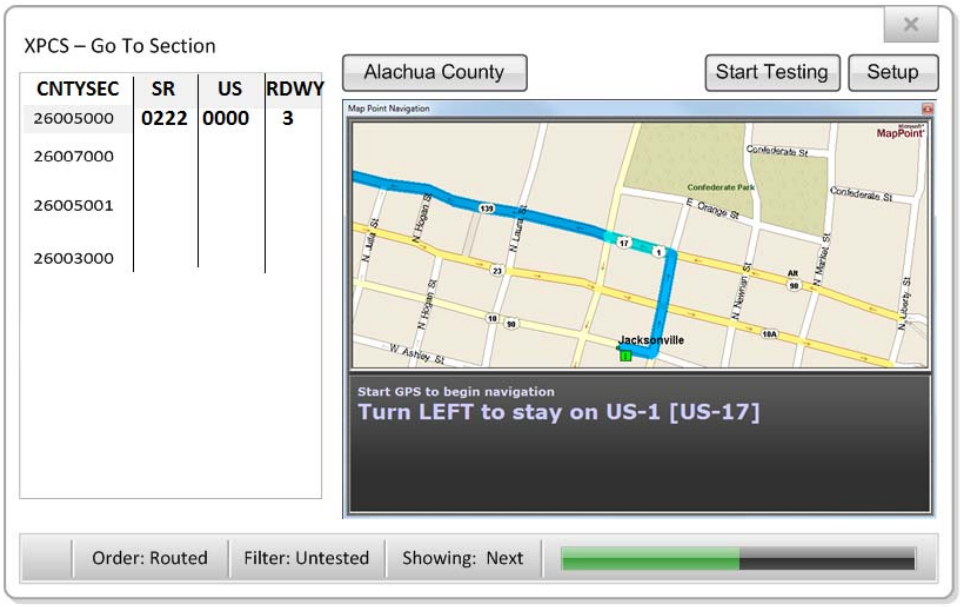

Figure B5. Navigation to Test Sections

Table B4. Navigation interface requirements

#	Description	Priority
Navigation Interface		
74	User is given audible announcements of directions to test sections.	C
75	User is given audible announcement of arrival at test section.	C
76	User can repeat the current section.	C
77	Application automatically tracks (via GPS) current segment.	C
78	User can specify navigating to any section including those that have been tested.	C
79	User can use MapPoint control to navigate sections.	D
80	User can turn off/on automatic section tracking.	C
81	Display the current progress of testing on a section-by-section basis.	D
82	Display overall testing progress (as a percentage of total sections for county).	C
83	Track van location.	C
84	Next test section will be visually identifiable on the GIS map. (Must be unique visual identifier, distinct from other begin test section locations).	C
85	All test sections will be visually identifiable on the GIS map.	C
86	End of current test section will be visually identifiable on the GIS map (Must be different identifier from other end section points).	C
87	User must be able to turn off/on visual identification of test sections.	C
88	Interface must show current status of test sections.	C
89	User will be able to manually reorder test sections.	C

Test Interface

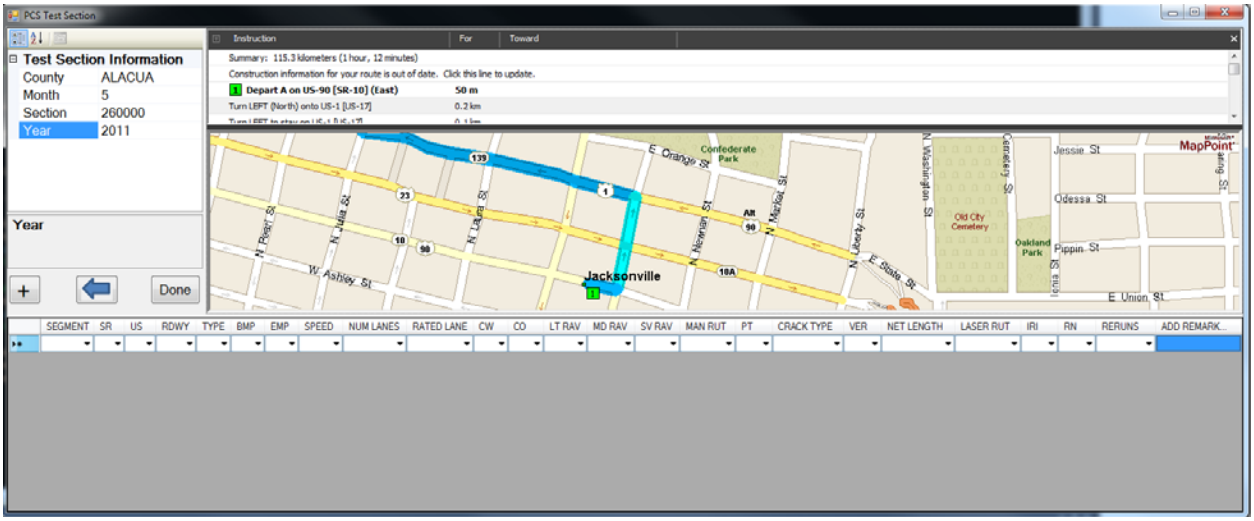

Figure B6. Interface during testing flexible pavement

The interface shown in Figure B6 is what the user sees during testing. There are three primary areas on the Test Interface. The first is a data grid of current test section data broken down by segment. The user can add/modify this data during the testing process. The property grid in the upper left pane shows data that is consistent throughout the test section. This data may also be modified if necessary. The third pane shows a map which updates according to the location of the profiler. Done and back (represented by the blue arrow) buttons take the user to the Validation and Navigation interfaces, respectively. Data modified on this screen is automatically committed to the database when the "Done" button is clicked.

The "+" button can be used to create a breakpoint in the test section so that a new segment can be added. The operator will input the milepost of the breakpoint when clicking the "+" button. All pertinent data will be copied into the new segment from the current segment.

Note that this interface is not designed to show prior year's data. ARA has assumed that the users will not want to see the data until the validation phase. This screen can be used at the conclusion of testing if entering data during the run is not safe or effective. The software will move to the validation phase only when the operator clicks "Done".

A similar interface will be developed for evaluating rigid pavements. Functional requirements for all testing interfaces are shown in Table B5.

Table B5. Testing interface requirements

#	Description	Priority
Testing Interface		
90	Allow entry of breakpoint. Operation creates a new segment populated with data from the existing segment.	C
91	Allow user to change segment regardless of vehicle location.	C
92	Change segment automatically based on current location.	D
93	Interface automatically adjusts depending on pavement type (rigid or flexible).	C
94	Allow user to turn off/on automatic segment tracking.	D
95	User can commit segment data to the database.	C
96	User can navigate to test segments from MapPoint control.	D
97	User needs to be able to delete segments.	C

Validation Interface

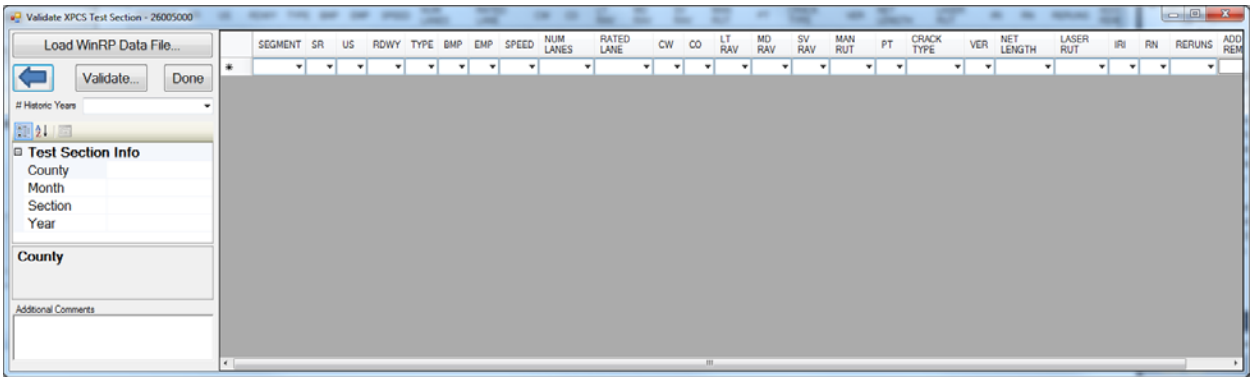

Figure B7. User Validating Test Section Data

The validation interface, as shown in Figure B7, is the final screen an operator uses prior to completing a particular section. The "Done" button in this case will return the user to the Navigation Interface if all issues are resolved. The user may force the change to the Navigation Interface from this screen, but the current section will be flagged with a status of "Incomplete". The user may also return to the testing interface to (re)enter condition or roadway data by using the back button.

This interface allows the user to load data from a WinRP text file and compare all condition data (automated and manual) with the prior year results. Clicking the "Validate…" button will run all automated validation checks against the collected data. The data grid will display all of the segments for a given section. Each segment in the data grid is represented (by default) by four rows. The first row represents the data just loaded from the WinRP file. The three rows following represent the historic data for that segment three years back (one row per year). Users can alter the number of historic rows to view by changing the drop-down box entitled "# Historic Years". The current and historic data rows will be shaded different colors for ease of use. WinRP files only need to be loaded once (as opposed to

once per section). WinRP files are loaded into the PCS database. The load routine will recognize the locations from the WinRP file and place the ride and rutting data into the appropriate segments.

Differences that fail validation checks are highlighted in red. Segments are highlighted in red on the left list when they do not contain all required data. The operator may return to the testing interface to provide the missing data. Segments highlighted in yellow on the left list have all required data but some of that data has failed validation checks. A test section may be marked as "complete" with validation errors. However, the user must enter a comment in the "Comments" field for the segment regarding these failures. The Additional Comments text box is used to enter free-form comments regarding the test section as a whole. Standard comments are entered through the test interface. A full list of functional requirements is provided in Table B6.

Table B6. Validation interface requirements

#	Description	Priority
Validation Interface		
98	Display segments with missing data.	C
99	Display data points that have failed validation checks.	C
100	Require user to input comments on a section if a segment fails validation checks.	C
101	A status must be associated with each test section. (See Requirement #98.)	C
102	Allow users to input comments on all test sections.	
103	Change status of test sections. Acceptable status includes: • Complete (Green) • Complete (Validation Errors) – This status requires comments (Blue) • Data Incomplete – Incomplete data (Yellow) • Validation Incomplete – Validation error with reason. NOTE: This requirement implies that given a reason or comment for validation failure a section can be marked as complete (Red). • Untested (Grey).	C
104	Allow administrative users to change validation thresholds.	D
105	Visual indication of status must be presented to the user in this interface. This status must reflect the color coding scheme used for sections in the Navigation Interface.	C
106	Data loaded from WinRP is stored in the PCS database.	C
107	Allow users to view 3 previous years data for a specific segment (3 years should suffice).	C

Hardware Interfaces

There are a few requirements that also must be met on the hardware side of the project based on FDOT's current equipment. Those requirements are listed in Table B7.

Table B7. Hardware interfaces

#	Description	Priority
Hardware Interfaces		
108	Existing profiler computer with Windows XP Service Pack 2.	C
109	GPS unit proven to work with profile computer.	C
110	High speed profiler.	C

Other Requirements

Finally, there are several other miscellaneous requirements that do not properly fit within the previous sections of the report. These requirements include those required for proper software integration, those required to meet FDOT IT requirements, and items relating to existing PCS data. These requirements are shown in Table B8.

Validation Rules

Existing rules from the excel spreadsheet will be applied in an intuitive manner in the application's data validation interface.

Data rows (segments) not matching specific validation thresholds defined by the user in the validation interface will be highlighted with a red background. Before synchronization can occur, all data validation rules must pass or comments must be entered. Clicking on a row will give the user more detailed output as to why the row or highlighted cell failed validation.

Future Work

Current integration with WinRP can be improved. Working with ICC, ARA would like to work to improve the functionality between the two systems by automatically pulling in WinRP test section data at the end of a run. ARA would also like to work further with ICC to automatically detect the begin and end of test sections, and to have the WinRP files auto-generate from pre-populated templates for those sections.

Table B8. Other requirements

#	Description	Priority
Software Requirements		
111	Latest version of WinRP (WinReport) in use at FDOT (version 2.1.2.1).	C
112	INI file designed to create a WinRP report compatible with the import process in the software validation interface.	C
113	SQL Server Express 2008.	C
114	.NET Framework 4.0 or better.	C
115	Microsoft MapPoint.	C
IT Requirements		
116	FDOT IT allows for connected synchronization procedures to occur.	D
117	Operators must have sufficient privileges to perform database synchronization.	C
Data Capacity		
118	Onboard database must not exceed 10 GB in file size.	C
Data Retention		
119	Data will be stored in the PCS Database until archived according to FDOT procedures.	C
Data Conversion		
120	All historic data will be migrated from the RCI mainframe database into the PCS database.	C
121	Pertinent PCS data will be synchronized with the van database.	C
122	Optimized routing data will be stored in the PCS database.	C
123	If possible, GPS data pertaining to sections and segments will be stored in the PCS database.	C
Error Handling		
124	Application will maintain robust error handling.	C
125	Users will be informed of system issues and errors.	C
126	Application must log crashes in an intuitive fashion.	C
127	Application must be able to “rollback” synchronization in the event of a system or network error.	C

Appendix C

Software Design Document: Office Component

Table of Contents

List of Figures

List of Tables

Introduction

This document will discuss the detailed design and architecture of the Office XPCS system.

Overview

The Office XPCS system is being developed for the FDOT State Materials Office (SMO). The system is comprised of three main components, the ArcGIS plug-in utility, XPCS Database Synchronizer (referred to as the Synchronizer), and the Office XPCS Application.

The first component is the ArcGIS plug-in tool. This tool shall be created within ArcGIS as a plug-in and will accept shapefile data for transferring test section location information between ArcGIS and the XPCS database. The synchronizer serves as a process to maintain data integrity in a disconnected data environment. The application itself contains PCS validation, verification, editing, visualization, and reporting features. These features support the FDOT personnel during their yearly update of the PCS test sections.

The application will run on Windows XP SP2 (or higher), or Windows 7 machines in the Florida State Materials Office (SMO).

Scope

The Office XPCS system has three primary design goals. The first goal is to reduce the amount of overhead in transferring, validating, and correcting collected PCS data. The second goal is to reduce and eventually eliminate the need for SAS interaction with PCS data and the third and final goal is to generate information rich reports on PCS survey data and status. These goals are to be achieved while maintaining application scalability for future work, and expanding usage of the application to multiple FDOT survey vehicles.

Background

For a detailed background of the project, please reference the FDOT Office XPCS Requirements Document.

Assumptions

FDOT and ARA anticipate the usage of this application to spread to multiple FDOT vehicles. This assumption will be paramount throughout the design process.

Constraints

Due to the size and complexity of ArcGIS Network Analyst, ARA will not be providing a rich interaction with Network Analyst as a part of the overall design of the Office XPCS application. Instead ARA will, during the implementation phase, provide training to select FDOT employees on using Network Analyst with the system. After the training has occurred, it will be the responsibility of FDOT to “fine tune”

Network Analyst to produce the optimum routes for survey vehicles.

Requirements Compliance Matrix

Table C1 shows the current requirements and describes how the proposed design implements those requirements. Additionally, this table can be used to describe and track changes made to the requirements throughout the development process.

Table C1. Requirements compliance matrix

#	Description	Compliance	Assumptions
Functional Requirements			
1	GIS plug-in must export test section locations into PCS database in order of their sequence number produced during the optimization process.	The GIS plug-in populates the RouteSections table of the XPCS database with the appropriate section and sequence number.	
2	Validate section/segment limits against the RCI mainframe.	The RCIUpdater class in the XPCSData module checks and optionally updates section/segment limits against the RCI database.	
3	Validate jurisdiction for each section against the RCI mainframe.	The RCUpdater class in the XPCSData module checks the jurisdiction automatically.	
4	Store section spatial/geometric data.	The Sections and Segments tables contains spatial data. The spatial can be retrieved from any feature class.	Spatial data for sections will be made available in the form of shapefiles.
5	Store section begin/end locations as GPS data.	The Sections and Segments tables contain spatial data.	Spatial data for sections will be made available in the form of shapefiles.
6	Allow simple (closest section) optimization of test sections.	The ArcGIS Network Analyst allows the user to export optimized section ordering to the XPCS database.	
7	Allow multiple attribute optimization of test sections.	Currently, the ArcGIS section optimization occurs only on a single variable.	
8	Allow users to modify all collected data.	The data management user interface allows the user to modify inspection data.	
9	Allow users to modify section/segment limits.	The data management user interface allows the user to modify network definitions.	
10	Automation of all ArcGIS processes for generating optimal routes.	Currently, the ArcGIS plug-in only handles the export of the results of the optimization process.	
11	“Dashboard” view of % sections completed.	The Dashboard class calculates and displays the appropriate data.	
12	“Dashboard” view of % sections on time.	The Dashboard class calculates and displays the appropriate data.	
13	“Dashboard” view of % sections validated.	The Dashboard class calculates and displays the appropriate data.	

Table C1. Requirements compliance matrix (con't)

#	Description	Compliance	Assumptions
14	GIS layer showing testing progress.	The Test Section GIS interface provides this information to the user using SharpMaps.	
15	GIS map allows users to select sections to generate section status reports.	The Test Section GIS interface allows the user to select a section and generate section specific reports.	
16	Must generate section validation report against the RCI database.	This report class uses the RCUpdate results to display this data to the user.	
17	Must generate data validation report against historical (previous years) data.	The validation report is implemented.	
18	Must generate Flexible PCS and Rigid PCS reports from the PCS database.	These reports are implemented in the Reporting module.	
19	Must be able to define data validation thresholds.	Validation rules are customizable, either as plain-text or through a validation wizard.	
20	Data validation thresholds must be pushed out to mobile XPCS databases.	Data validation rules are stored in the database as plain-text that is synchronized with the mobile databases.	
Login Interface			
21	Maintain user name across sessions.	"Last-good" settings are stored in an initialization file once the user successfully logs in.	Only save credentials that have been successful.
22	Maintain database connection information across sessions.	"Last-good" settings are stored in an initialization file once the user successfully logs in.	Only save credentials that have been successful.
23	Allow users to set up database connections to mobile XPCS databases.	The user has the option to (attempt to) connect to any SQL Server database.	
24	Provide users with information rich feedback on login/setup issues.	Any errors will be presented to the user with the most information available.	
Security/User Management Interface			
25	Manages two types of account access: users and administrators.	The Security module can manage an arbitrary number of user roles.	Only two roles will be setup during implementation (User and Administrator)
26	Application must allow for the creation of new accounts.	The SecurityManager module allows new users to be entered into the SecUsers table.	
27	Application must allow for the deletion of accounts.	The SecurityManager module allows users to be removed from the SecUsers table.	

Table C1. Requirements compliance matrix (con't)

#	Description	Compliance	Assumptions
28	Application will allow for cancelling changes made.	No changes to security settings can happen until the user has confirmed them.	
29	Administrators will have access to all features and functions in the application.	By default, the SecurityManager maintains an "Administrators" user group that has the highest possible permissions level for the "All Actions" action group.	
30	Users will be able to modify, validate, and synchronize PCS data only.	Permissions can be controlled at any level of granularity, but the default implementation will provide these constraints to the "Users" usergroup.	
Dashboard Interface			
31	Shows instant report of sections tested.	The Dashboard class calculates and displays the appropriate data.	
32	Shows instant report of sections validated.	The Dashboard class calculates and displays the appropriate data.	
33	Shows instant report of sections completed on time.	The Dashboard class calculates and displays the appropriate data.	
34	Report can be refreshed on demand.	The Dashboard class periodically refreshes against the database.	
35	Data is updated from the PCS database.	The Dashboard class periodically refreshes against the database.	
36	Can be easily viewed/hidden.	The Dashboard form will minimize with a double-click.	
Reporting Interface			
37	Allow users to select from various reports.	The reporting user interface allows the user to view any of the available report types.	
38	New reports can be added to the Reports Management interface.	The reporting user interface allows the user to add custom reports via Crystal Reports.	
39	Generating a report runs custom report code to create the currently selected reports.	Removed due to redundancy with requirement #38.	
40	Interface must allow users to cancel report generation and return to main screen.	Every report invokes a loading screen that provides a cancel button.	
41	Flexible PCS report converted from SAS to SQL Server 2008	Flexible PCS report implemented.	
42	Rigid PCS report converted from SAS to SQL Server 2008	Rigid PCS report implemented.	

Table C1. Requirements compliance matrix (con't)

#	Description	Compliance	Assumptions
43	Lane Miles surveyed on the state maintained roadway system report converted from SAS to SQL Server 2008 (two types).	Lane miles report implemented.	
44	RCI Verification report converted from SAS to SQL Server 2008.	Verification report implemented.	
Data Management Interface			
45	View PCS data in format similar to existing worksheets in accordance with FDOT standards (Highlighting rules given in FlexWorkbookPointofEntryChecks.doc).	The data management user interface applies the same validation checks/highlighting rules as in the mobile application.	
46	Modify PCS data.	All data will be available for editing in this user interface.	
47	Validate selected rows of PCS data.	Data is automatically validated and flagged upon being loaded into the data management interface.	
48	Validate all PCS data.	Data is automatically validated and flagged upon being loaded into the data management interface.	
49	Verify PCS jurisdiction and mileposts with RCI database.	The RCIUpdater class in the XPCSData module checks and optionally updates jurisdiction and section/segment limits against the RCI database.	
50	Verify selected rows with RCI database.	The RCIUpdater class in the XPCSData module checks and optionally updates jurisdiction and section/segment limits against the RCI database.	
51	Allow filters to be created for any of the PCS header data.	The Data Management user interface provides an advanced search feature.	
52	Allow filters to be saved.	The Data Management user interface allows the user to save advanced search queries.	
53	Allow filters to be loaded.	The Data Management user interface allows the user to load advanced search queries.	
54	Only allow changes to go through if all data is properly validated, verified and/or commented.	The validation rules prevent the Accepted flag from being set in either of the inspection tables.	
55	All changes made to be rolled back until "Submit Changes" is clicked.	Changes will be superficially stored in the database until this button is clicked.	

Table C1. Requirements compliance matrix (con't)

#	Description	Compliance	Assumptions
56	Allow users to see data for both flexible and rigid pavements.	The datagrid on the management interface is capable of displaying both flexible and rigid pavement data.	
57	Allow user to view historical data for selected sections.	Historical data for a user-specified number of years is automatically loaded along with the current data in the data management interface.	
Test Section GIS Interface			
58	Show all test sections in the state as line geometries.	The Test Section GIS interface will use SharpMap to automatically display this data to the user.	
59	Highlight test sections according to processing level. (Incomplete – Red, Tested – Blue, Validated – Green, Not Tested – Orange).	Each processing level has its own highlighting layer added to the SharpMap display.	
60	View section details from the GIS map.	The user is able to click on a section and view data from that section.	
61	Manage multiple layers in the GIS map.	The Test Section GIS interface provides a layer management form.	
62	Filter for desired sections based on county, section, subsection and mileposts.	The Test Section GIS interface provides the user with an advanced search capability that allows them to construct a query filter.	
Database Synchronizer			
63	Interface must allow users to input local server name (the "from computer").	The XPCS synchronizer allows the user to sync any number of XPCS databases at the user's selection.	
64	Interface must allow users to input local database instance name (the "from computer" database name).	The XPCS synchronizer allows the user to sync any number of XPCS databases at the user's selection.	
65	Interface must allow input of central server name (the "to computer").	The XPCS synchronizer allows the user to sync any number of XPCS databases at the user's selection.	
66	Interface must allow input of central server instance name (the "to computer" database name).	The XPCS synchronizer allows the user to sync any number of XPCS databases at the user's selection.	
67	The synchronizer must synchronize data between the from and to databases using a last modified data flag for updatable data.	The XPCS synchronizer allows the user to sync any number of XPCS databases at the user's selection.	

System Description

The Office XPCS application allows users to view and edit survey data, synchronize with data collected in the field, propose optimum routing strategies for survey teams, and provides meaningful reports on collected PCS data.

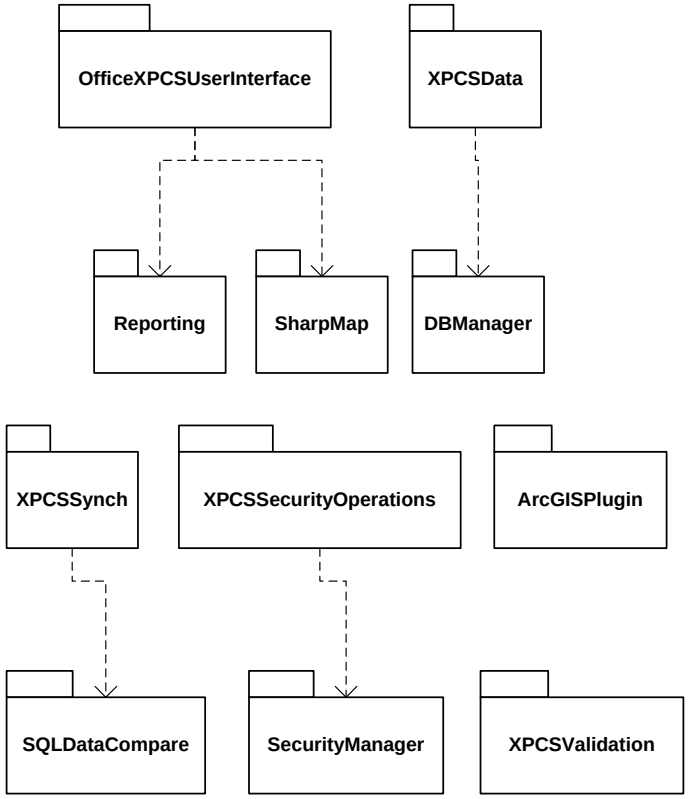

Figure C1. Office XPCS Modules

System Software Architecture

The Office XPCS system is broken down into several major software modules as illustrated in Figure C1. These software modules are briefly described in the sections below, and are further detailed in the Technical Specifications section of this document.

OfficeXPCSUserInterface

The Office Extended Pavement Condition Survey User Interface module houses the form classes and serves as the program entry point. The interfaces described in the SRD will be housed within this module.

Reporting

This module will utilize Crystal Reports to generate the three standard reports described in the Office SRD. In addition, this module will provide support classes for the Dashboard user interface element.

XPCSData
Houses the various factory classes that will create business logic data structures out of the database.

DBManager
This contains the generic database wrapper class used for all database calls and the SQL Server specific implementation of that class used in this system.

XPCSSync
This library provides the XPCSSynchronizer class that logically merges the databases from the collection vans with the central database used by the office application.

SQLCompare
Comparison classes for identifying differences between databases and producing SQL scripts to run on those databases in order to correctly merge the information contained within.

XPCSSecurityOperations
Provides static classes that hold the specific security check functions for the user interface to identify the type of user/action taking place in the system.

SecurityManager
Generic security system that provides an easy and flexible method of assigning users to roles, grouping actions and specifying permissions at any level of user/usergroup to action/actiongroup.

ArcGIS Plug-In
This module will contain the plug-in classes and functions which generate the optimal section ordering. The module will also create data structures to handle connecting to and moving optimized route data to the XPCS database.

XPCS Validation
Provides the business logic objects that compose the validation engine.

FDOT Survey Vehicle
The survey vehicle contains the FDOT vehicle computer, Microsoft puck, and the most important piece of the software equation…the surveyor!

Microsoft GPS Puck
The GPS puck will be used to determine the current location of the survey vehicle throughout the testing week.

FDOT Vehicle Computer
The computer which resides on the FDOT survey van. This computer will be responsible for running the mobile XPCS application, and interfacing with the GPS puck via USB plug-in.

USB Key
The USB Key is used by the field crew surveyor to store backups of the XPCS database and other

pertinent test section information. The Key is also used to transfer the XPCS database to and from the office for synchronization.

External Interfaces

This section will describe interfaces with other software applications or modules.

SQL Data Compare

Developers will work with the SQL Data Compare software development kit (SDK) in order to perform the synchronization of the mobile XPCS database and the office XPCS database. ARA has already acquired licenses for working with and distributing applications which utilize this SDK.

SharpMap

ARA has extensive experience working with the SharpMap open source GIS SDK. SharpMap's primary use will be in the TestSectionGISInterface, a part of the Office XPCSUserInterface module. SharpMap will provide the Office XPCS application with most of the functionality required by that interface, including the ability to query and overlay GIS data. The SharpMap source code can be made available upon request from FDOT personnel.

ArcMap

ESRI's ArcMap Network Analyst will fulfill the optimized route requirement for the Office XPCS application. Network Analyst will use the vehicle routing problem module to solve for optimized routes based off of a single starting depot (most likely the hotel starting point for the week), ordered pairs of test section starts and ends, and a return location. More information regarding the details of Network Analyst can be found at http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=Solving_a_vehicle_routing_problem.

Microsoft MapPoint

The Mobile XPCS application will utilize the Microsoft MapPoint 10 SDK to fulfill several user interface requirements with regards to driving directions, test section/segment locations et all. MapPoint will not be utilized in the Office XPCS application.

EasyGIS

EasyGIS is an open source GIS SDK that will be utilized to provide shapefile output from the Office XPCS system. The EasyGIS classes will need to interact with the XPCSData and OfficeXPCSUserInterface modules to achieve desired functionality.

Design Approach

This section will detail the order of, approach to, and completion milestones for each of the modules described earlier in this document and the approximate amount of time it will take to develop each module and subsystem.

Please note that several modules, once created for the Office XPCS application, will be used in totality in the Mobile XPCS application, and in the Mobile XPCS Software Design Document are listed with minimal

or no development time. These modules include, XPCSData, DBManager, XPCSSecurityOperations, SecurityManager, XPCSValidation. The XPCS system will reside within a single solution file, allowing us to take advantage of the many duplicate requirements between the two applications. Two separate executables will be generated from the single solution, one for the Mobile XPCS Application and one for the Office XPCS Application.

Total Development Time: 66 days.
Milestone: Completed development of all modules. Able to send application for user testing to FDOT.

Prototype ArcGIS Plug-In

Development Time: 5 days.
Milestone: Successfully export optimized route layer into XPCS SQL database.

Development of the ArcGIS plug-in will be done first as it contains most of the "uncertainty" in the project design. Completion of the ArcGIS plug-in will allow the user to export an optimized route list from Network Analyst into the XPCS database.

Develop XPCS Database

Development Time: 10 days.
Milestone: Successfully import historical PCS data (already provided by FDOT) into the XPCS database).

Before any of the other modules can really be developed, it will be important to have a working data set for testing purposes in place. This makes development of the XPCS database and the importing of the historical PCS data critical in the design path.

Develop DBManager

Development Time: 5 days.
Milestone: Completion of standard I/O database functions as they pertain to specific FDOT requirements.

The database manager contains classes and functions for storing, retrieving, and updating information contained within the XPCS system.

Develop XPCSSynch

Development Time: 10 days.
Milestone: Verify correct synchronization has occurred between two physically separate machines hosting the XPCS database.

The development of the synchronization module can be done in parallel with any other module development task after the XPCS database has been created and had historical data imported.

Develop XPCSData

Development Time: 1 day.
Milestone: Completion of underlying data access classes and successful testing of DAL through the

business logic with prototype user interface elements.

The XPCS Data Module contains all of the relevant data access classes and provides an object-oriented design for the database subsystem.

Develop OfficeXPCSUserInterface

Development Time: 15 days.
Milestone: Completion of all user interface elements as described in this document and the SRD. Successful user testing with FDOT personnel.

The user interface is the key for user interaction with the system. The development of this module will be very iterative with FDOT personnel, allowing them to pinpoint how the interface requirement features should be implemented.

Develop XPCSValidation

Development Time: 10 days.
Milestone: FDOT validation of the validation parameters as displayed in the user interface.

This module will again require close corroboration with FDOT personnel in order to correctly apply validation parameters in the user interface. This module contains all of the functions and classes necessary to validate user interface elements, including user input data and data loaded from files.

Develop SecurityOperations

Development Time: 5 days.
Milestone: Proper functioning of all user interface screens with both field users and administrators accessing the system.

The security operations module contains classes describing users, groups, actions and their various interrelations.

Develop Reporting

Development Time: 5 days.
Milestone: Successful run of required reports from the XPCS reporting interface. Allow addition of "sample" report developed outside the application.

Three major reports will be developed utilizing Crystal Reports in the Visual Studio IDE for FDOT. Future reports will be created by FDOT personnel and may be added to the report generating interface within the Office XPCS application.

Technical Specifications

This section will list the modules described above and go into further detail, describing subsystems to establish a strong reference with the Requirements Compliance Matrix.

OfficeXPCSUserInterface

This module contains all of the (non-reporting) form classes the user will interact with while using the office application as described in the Office XPCS Software Requirements Document and earlier in this document.

XPCSData

This is the factory floor for the majority of the business logic objects in the application. This module contains the data representation, and route classes that the rest of the application will use. An illustration of the objects that are members of this module are shown in Figure C2.

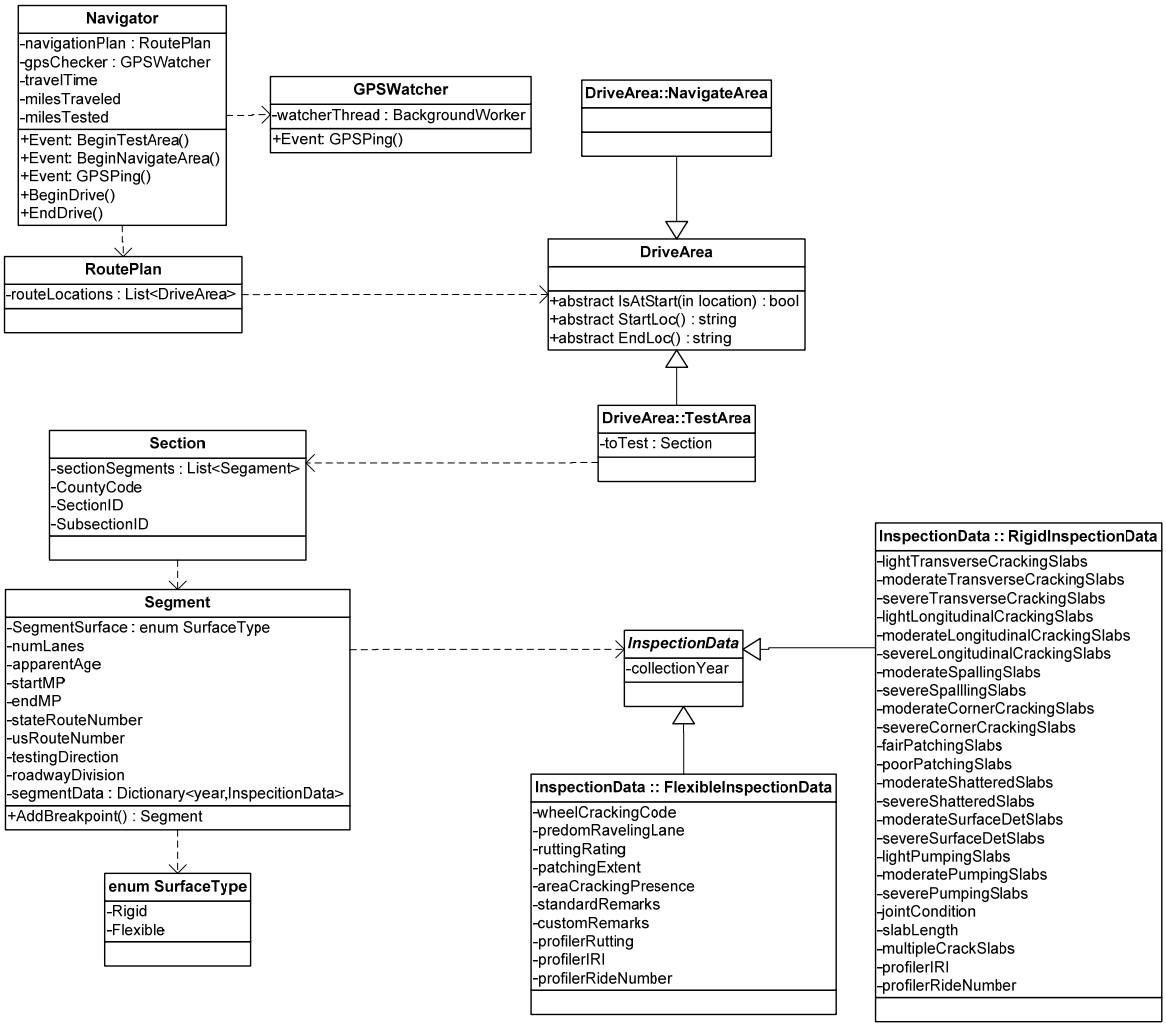

Figure C2. XPCSData module objects

DBManager

This module's primary purpose is to make the abstract Database class (and a SQL Server specific implementation class) available to the rest of the program. This class acts as a wrapper for database

calls and handles the grunt work of creating the connection, command, and adapter objects when queries are run.

XPCSSync

This module's primary purpose is to supply the XPCSSynchronizer class to be used by the synchronization forms for the purpose of merging mobile databases with the central database and with one another. This class makes use of the RedGate SqlCompare Application Programming Interface (API) that makes identifying inserted/deleted/updated rows very easy. With that part of the functionality provided for, the Synchronizer merely has to apply a "newest wins" rule and fix the tables up from the "bottom" (i.e., starting with the tables that have no foreign keys and then expanding to the tables that link to them and so on).

XPCSSecurityOperations

This module's primary purpose is to supply a static SecOps object to the user interface. The SecurityManager is a very general system that handles resolving a user specifier/action specifier pair into some permission level. The purpose of the SecOps object is to hold the context-specific blueprints for the action specifiers for each relevant security check. This allows for a highly granular breakdown of permissions as needed without cluttering user interface code with large amounts of action specifier code.

SecurityManager

The primary purpose of this module, illustrated in Figure C3, is to supply the SecurityManager class to be used by the SecOps layer. SecurityManager is responsible for resolving a descriptor of a specific user and a specific action into a permissions level (None, ReadOnly, ReadWrite, Create, CreateDestroy) against a set of (user/usergroup)->(action/actiongroup) permissions. Permissions are evaluated from least specific to most specific (with user specificity trumping action specificity) in the following way:

1. Usergroup->Actiongroup
2. Usergroup->Action
3. User->Actiongroup
4. User->Action

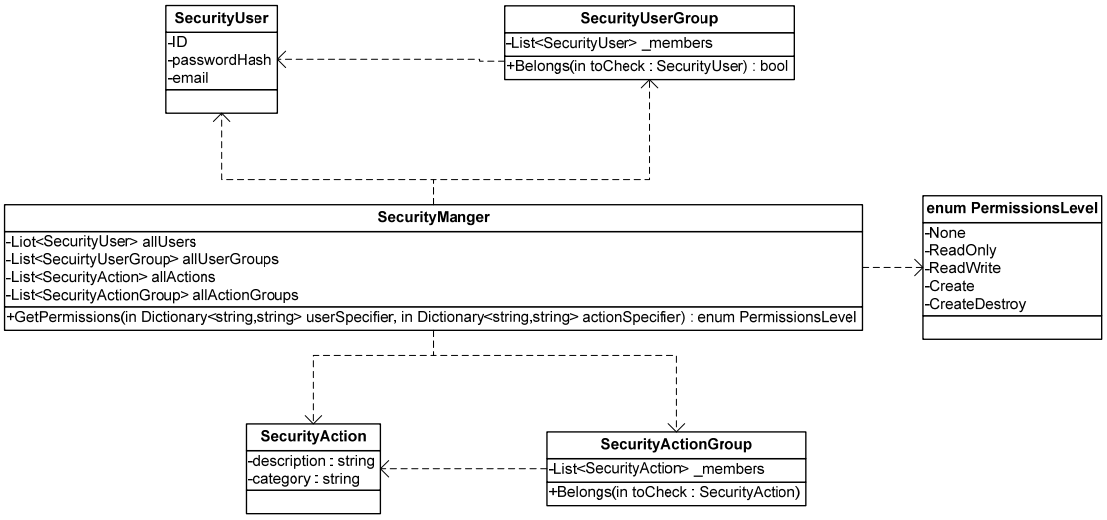

Figure C3. SecurityManager objects

XPCSValidation

This module supplies the RigidSegmentValidator, RigidSectionValidator, FlexibleSegmentValidator and FlexibleSectionValidator classes. These classes are used to verify the entry of data and flag whether or not the section needs to be retested. These classes implement the primary IValidator interface and are themselves composed of internal IValidator classes that check specific requirements derived from FDOT's documentation. An example of an existing data validation process currently used by FDOT is shown in Figure C4.

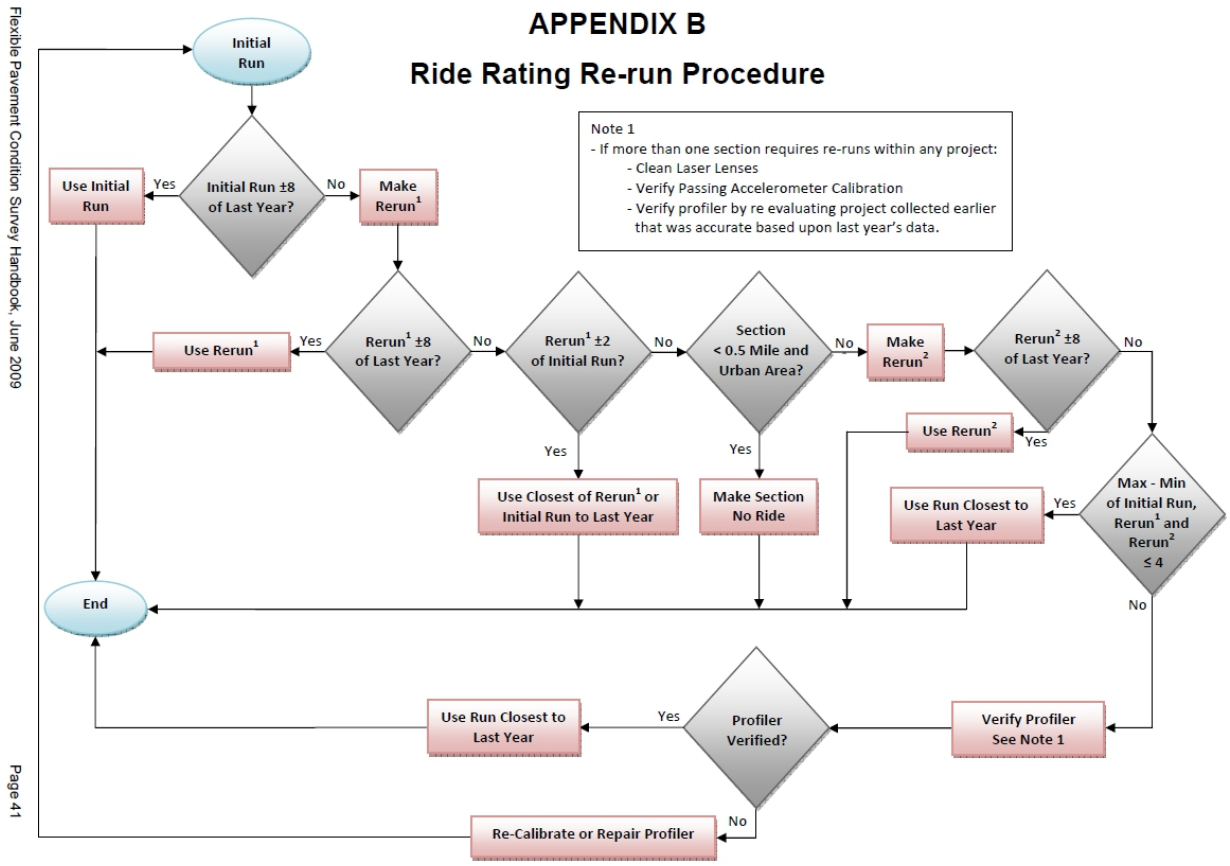

Figure C4. Ride rating rerun procedure

The validation classes all serialize into basic (human readable/editable) descriptors of input and rules (which can then be used by the ValidatorFactory to produce the current rule objects for checking each segment/section. The objects involved in the validation process are shown in Figure C5.

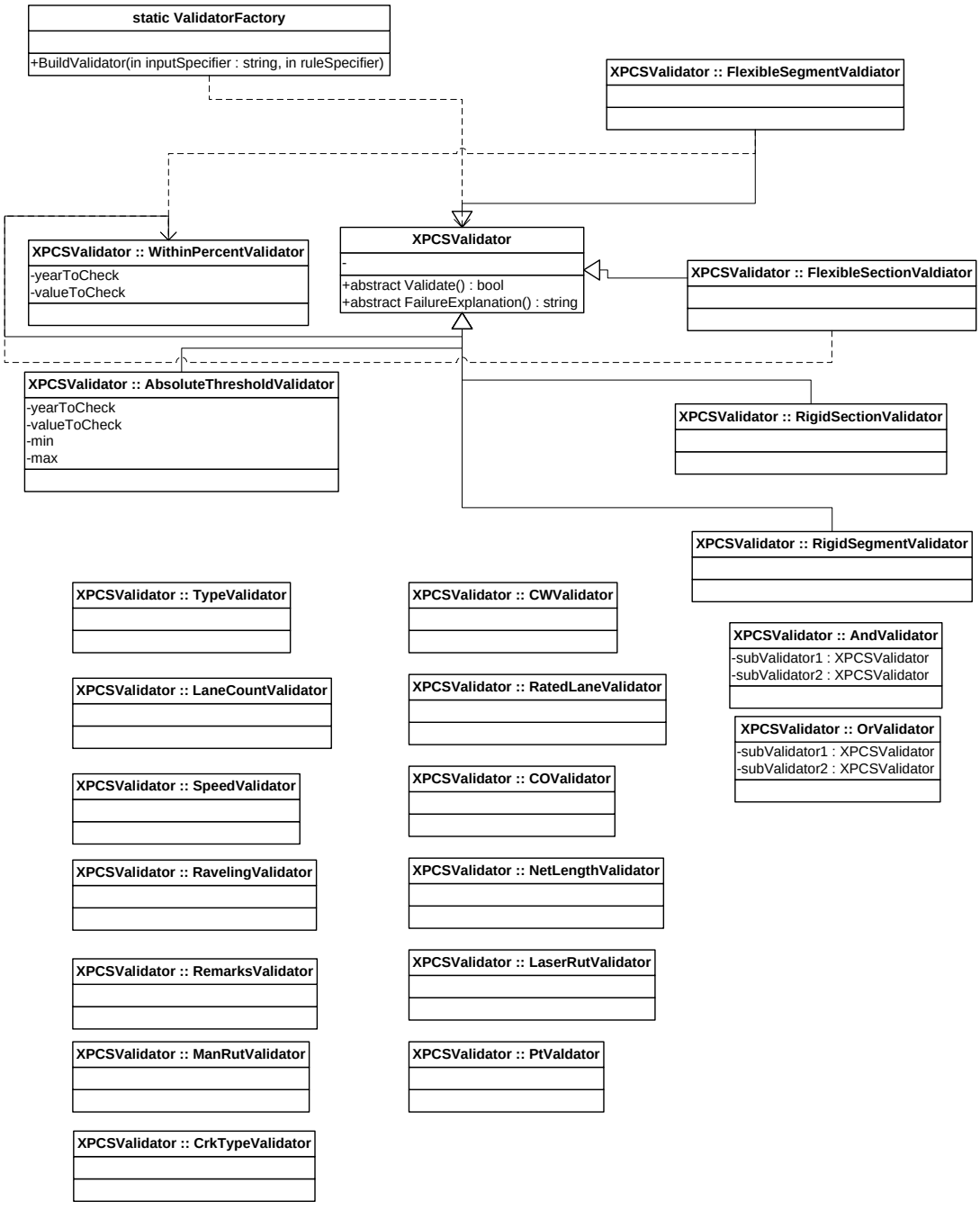

Figure C5. Validation model objects

Reporting

The reporting module will utilize interaction with Crystal Reports to generate the required reports as described in the Office SRD. Each of the three reports will be generated from the XPCS database. Classes will be developed based on interaction with the Crystal Reports SDK to appropriately manage the XPCS data into the final reports.

User Interface Design

The user interface design will be managed dynamically with FDOT staff, within the bounds of the interface requirements as described in the Office XPCS SRD.

Data Architecture

This section of the design will discuss the database schema and design decisions made regarding how data is stored in the XPCS system.

File Architecture

The Office XPCS system does not utilize any external files with the exception of configuration files for the application. These files allow users to modify initial application settings and are deployed as a part of the overall solution during installation.

Database Management System Architecture

The database schema used to represent the inspection data and security settings is shown in Figure C6. The individual objects within this schema are discussed in detail within this section.

Security Tables

SecUsers - Holds user information (id, password hash, email) for the security system.

SecActions – Holds action information (id, category, description) for the security system.

SecUsergroups – Lists the usergroups in the security system.

SecActiongroups – Lists the actiongroups in the security system.

SecUserMembership – Holds information about which users belong to which usergroups.

SecActionMembership – Holds information about which actions belong to which actiongroups.

SecUserActionPerms – Holds permissions information between single users and single actions.

SecUserActiongroupPerms – Holds permissions information between single users and groups of actions.

SecUsergroupActionPerms – Holds permissions information between groups of users and single actions.

SecUsergroupActiongroupPerms – Holds permissions information between groups of users and groups of actions.

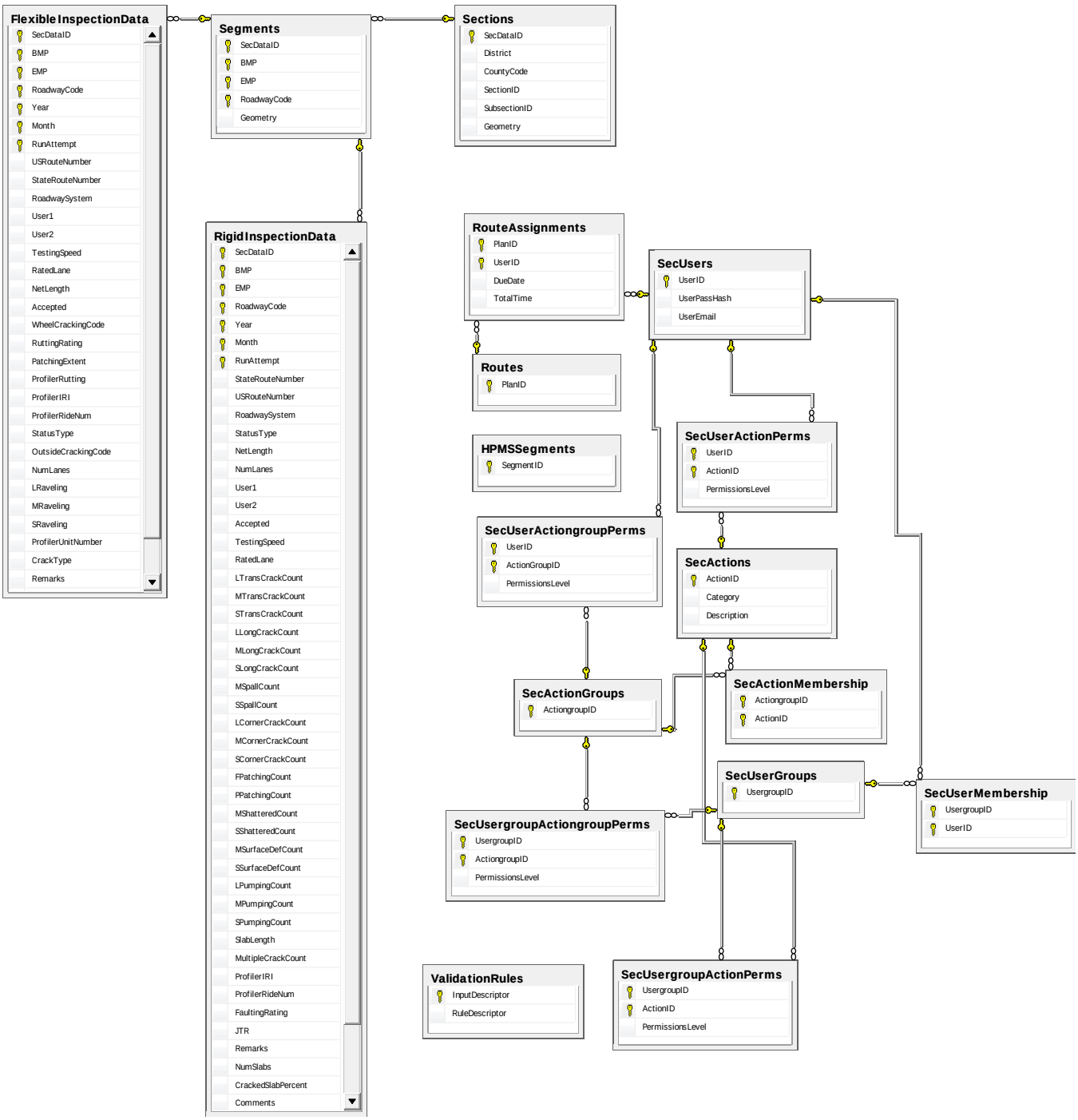

Figure C6. Overall database schema

Inspection Tables

Sections - This is the top level network table. It contains all possible test sections currently represented in the XPCS system.

Segments - This is the bottom level network table. It contains the segment information for the network. The roadway code is a part of the segment identifier.

FlexibleInspectionData/RigidInspectionData - These tables hold the segment-specific, year-specific, run-specific inspection data for their respective pavement types.

HPMSSegments – This table tells identifies sections that need to be tested for HPMS purposes.

Routes - When new optimized routes are inserted into the database, they are given a new key that resides in this table.

RouteSections -This contains the sections that make up each route and the optimal visit order.

RouteAssignments - This links generated routes with particular users so that they can pull up the routing information that is relevant. Additionally, it holds the due date (for scheduling purposes) and the total time spent surveying (for reporting purposes).

Validation Tables

ValidationRules - This table holds the plain-text serialized validation rules. Each rule is associated with a plain-text descriptor of the relevant datafield to which it applies. These strings are then used by the ValidationFactory to construct the proper validation objects.

Data Conversion

Currently existing survey data exist in the form of SAS datasheets. FDOT has provided a data dump of all their historical PCS data as Excel spreadsheets for inclusion in the XPCS database. ARA will develop a "one time application" export of the data into the XPCS database. No further data conversion is required.

Security

Security in the XPCS system is handled via the SecurityManager module. This module allows for the specification of usergroups and actiongroups. By default, there exist a usergroup called "administrators" and an actiongroup called "All Actions" which is automatically populated with new actions and over which they are given the highest level of access (Create/Destroy). While more groups are possible, FDOT's requirements only call for two levels of (non-administrator) permissions: office and field users. As the relevant security actions are discovered, they will be placed into actiongroups specifically set up for each of these usergroups. The user interface code can then check these permissions via the SecOps object and enable/disable user controls as required in order to present them with only valid options.

Capacity

The XPCS system has more pitfalls in terms of complexity in the business logic than in data capacity issues. The amount of data being collected and used is relatively small (in the requirements documentation, 10 GB is the upper limit given and that falls *far* above what is likely to be produced under any reasonable scenario). As such, we are free to use what would normally be bulk operations in order to overcome certain technical limitations of the system. For example, the (currently) disconnected nature of the data collection vans necessitates backing up the van database, copying it to a portable device (flash drive, etc.), and restoring it on a database server that can be reached by the XPCS office software in order to begin synchronization (and then reversing the process to get the synchronized data back to the van). If this system were more data intensive, this process would be onerous.

Performance and Timing

The Office XPCS database has been developed with performance in mind. However, performance is not critical to the operation of the application. Due to the small amount of data the application is required to handle and the performance boost given by utilizing an object-oriented "Active Record" design pattern, we expect performance to be at better than acceptable levels. It is much more important that the Mobile XPCS application have increased performance due to the live nature of the application. These performance goals will be realized within the Office XPCS Application due to the close nature of the two programs.

Error Handling

The Office XPCS application will utilize C# standard exception handling and will output data dump files to the program directory or users directory when application errors occur. This will help users pinpoint bugs and enable faster turnaround for managing bug resolution. In addition, the users will have access to the Mantis bug reporting system website for the XPCS application and will be able to report bugs and monitor their resolution in a timely manner.

Adaptability

In general, the system's adaptability comes primarily from the parameterization of the validation logic and the extensibility of the security system and the reports. The system should easily expand to further survey vans.

Appendix D

Software Design Document: Mobile Component

Table of Contents

List of Figures

List of Tables

Introduction

Overview

Mobile Extended Pavement Condition Survey (XPCS) is a software application being designed for the purpose of improving the data collection process of employees of the Florida Department of Transportation. The improvements focus primarily in three areas: routing/navigation, validation/verification, and modernization/efficiency.

Scope

The mobile component of the XPCS system is tasked with tackling the ground-level problems of data collection for FDOT's pavement condition surveys. Specifically, this means the application is designed to facilitating the survey crews getting to the test sections, entering manual and automated survey data into an onboard, disconnected database, and verifying that data against a set established rules.

Although there are data structures common to the mobile and office components of the XPCS system that naturally lead toward a shared library design, there are higher level concerns that are explicitly not within the scope of this portion of the system, but are instead left to the office component. Specifically, the mobile component is not responsible for the generation of optimized test section ordering (but rather navigation to and between sections), synchronization of the mobile databases with the central database, or the generation of reports from the central database.

Requirements Compliance Matrix

This table represents current requirements and describes how the proposed design implements them. Additionally this table can be used to describe and track changes made to the requirements throughout the development process.

Table D1. Requirements compliance matrix

#	Description	Compliance	Assumptions
Section Identification			
1	Reduce the time and complexity of identifying the start and end of a test section.	This requirement is met by the Navigator object and the MapPoint interface. MapPoint provides the routing to the start and end of each section and the Navigator polls the GPS device and fires the relevant event to update the MapPoint interface.	
2	Pull all pertinent inspection information for test sections the surveyor plans to evaluate in a specific trip into the vehicle database	As part of the synchronization process, section inspection data is automatically pulled in.	
3	Import optimized routing instructions	As part of the synchronization process, the optimized section ordering for a particular run is automatically pulled in.	

Table D1. Requirements compliance matrix (con't)

#	Description	Compliance	Assumptions
4	Display optimized routing instructions in MapPoint.	Once the user selects a run, the Navigator object uses the optimized section ordering to create a RoutePlan object and feeds the destinations to MapPoint	
7	Display current county	MapPoint provides this functionality directly.	
8	When traveling to a test section, display the next test section's ID and starting location	The Navigation user interface directly shows the ID of the section being navigated to and the start location on the map.	
9	When testing, display the test section ID and the end location	The Testing user interface directly shows the current section ID and the end location on the map.	
10	When testing, allow the operator to increase and decrease the segment	The testing user interface provides the user a checkbox to turn on and off auto-following and automatically turns off if the user selects a new segment	
11	Suggest current segment based on GPS location	The testing user interface presents the user with a segment list and attempts to track down the list automatically (indicating the start and end points of the current segment on the map).	
12	Display & edit the State Route Number	The testing user interface allows the user to edit this value in the Sections table.	
13	Display & edit the US Route Number	The testing user interface allows the user to edit this value in the Sections table.	
14	Display & edit roadway testing direction (ascending/descending)	The testing user interface allows the user to edit this value in the RouteSections table.	
15	Display & edit roadway division (composite/divided)	The testing user interface allows the user to edit this value in the Sections table.	
16	Display & edit section status	The testing user interface allows the user to edit this value in the inspection tables.	
17	Display beginning milepost for current segment	The testing user interface shows this data.	
18	Display ending milepost for current segment	The testing user interface shows this data.	
19	Add breakpoint at user defined milepost	The testing interface allows the user to insert a breakpoint (splitting a segment in two) at any point in the section.	
20	Add breakpoint at current location	Because of the difficulty in getting the current milepost from just the GPS information, the current design does not attempt to satisfy this requirement.	
21	Display & edit testing speed	The testing user interface allows the user to edit this value in the inspection tables.	
22	Display & edit number of lanes	The testing user interface allows the user to edit this value in the Segments table.	

Table D1. Requirements compliance matrix (con't)

#	Description	Compliance	Assumptions
23	Display & edit rated lane	The testing user interface allows the user to edit this value in the inspection tables.	
24	Display & edit pavement type	The testing user interface allows the user to edit this data in the Segments table.	
25	Change interface based on type of pavement: rigid or flexible	The data entry grid of the testing interface changes automatically based on this field.	Pavement type does not change mid-segment
Flexible Surface Condition Inspection			
26	Input observed wheel path cracking code	The flexible pavement layout of the data entry grid on the testing user interface provides a field for this value to be edited in FlexibleInspectionData table.	
27	Input observed non-wheel path cracking code (outside wheel path)	The flexible pavement layout of the data entry grid on the testing user interface provides a field for this value to be edited in FlexibleInspectionData table.	
28	Input the predominant raveling level (may be none)	The flexible pavement layout of the data entry grid on the testing user interface provides a field for this value to be edited in FlexibleInspectionData table.	
29	Input the raveling extent (if raveling level > none)	The flexible pavement layout of the data entry grid on the testing user interface provides a field for this value to be edited in FlexibleInspectionData table.	
30	Input a manual rutting rating (only after completion of run)	The flexible pavement layout of the data entry grid on the testing user interface provides a field for this value to be edited in FlexibleInspectionData table.	
31	Input patching extent	The flexible pavement layout of the data entry grid on the testing user interface provides a field for this value to be edited in FlexibleInspectionData table.	
32	Input area cracking presence (may be none, alligator, block, or both)	The flexible pavement layout of the data entry grid on the testing user interface provides a field for this value to be edited in FlexibleInspectionData table..	
33	Display verification flag	All data passes through the validation engine as soon as it is entered into the system.	
34	Display & edit standard remarks	The testing user interface allows the user to edit these values in FlexibleInspectionData table.	
35	Display net length (length of section minus any length not recorded due to roughness off/on flags)	The testing user interface shows this data from the FlexibleInspectionData table.	

Table D1. Requirements compliance matrix (con't)

#	Description	Compliance	Assumptions
36	Display profiler rutting	The testing user interface shows this data from the FlexibleInspectionData table.	
37	Display profiler IRI	The testing user interface shows this data from the FlexibleInspectionData table.	
38	Display profiler Ride Number	The testing user interface shows this data from the FlexibleInspectionData table.	
39	Display previous year values for items 26-32 and 34-38	The testing user interface shows this data from the FlexibleInspectionData table.	
40	Input free-form comments	The flexible pavement layout of the data entry grid on the testing user interface provides a field for this value to be edited in the FlexibleInspectionData table.	
Rigid Surface Condition Inspection			
41	Input number of slabs affected by transverse cracking (light, moderate, and severe)	The rigid pavement layout of the data entry grid on the testing user interface provides fields for these values to be edited in the RigidInspectionData table.	
42	Input number of slabs affected by longitudinal cracking (light, moderate, and severe)	The rigid pavement layout of the data entry grid on the testing user interface provides fields for these values to be edited in the RigidInspectionData table.	
43	Input number of slabs affected by spalling (moderate and severe)	The rigid pavement layout of the data entry grid on the testing user interface provides fields for these values to be edited in the RigidInspectionData table.	
44	Input number of slabs affected by corner cracking (moderate and severe)	The rigid pavement layout of the data entry grid on the testing user interface provides fields for these values to be edited in the RigidInspectionData table.	
45	Input number of slabs affected by patching (fair and poor)	The rigid pavement layout of the data entry grid on the testing user interface provides fields for these values to be edited in the RigidInspectionData table.	
46	Input number of shattered slabs (moderate and severe)	The rigid pavement layout of the data entry grid on the testing user interface provides fields for these values to be edited in the RigidInspectionData table.	
47	Input number of slabs with surface deterioration (moderate and severe)	The rigid pavement layout of the data entry grid on the testing user interface provides fields for these values to be edited in the RigidInspectionData table.	
48	Input number of slabs affected by pumping (light, moderate, and severe)	The rigid pavement layout of the data entry grid on the testing user interface provides fields for these values to be edited in the RigidInspectionData table.	

Table D1. Requirements compliance matrix (con't)

#	Description	Compliance	Assumptions
49	Input joint condition	The rigid pavement layout of the data entry grid on the testing user interface provides a field for this value to be edited in the RigidInspectionData table.	
50	Input estimated slab length	The rigid pavement layout of the data entry grid on the testing user interface provides a field for this value to be edited in the RigidInspectionData table.	
51	Input estimated number of slabs with multiple cracks	The rigid pavement layout of the data entry grid on the testing user interface provides a field for this value to be edited in the RigidInspectionData table.	
52	Input estimated number of slabs	The rigid pavement layout of the data entry grid on the testing user interface provides a field for this value to be edited in the RigidInspectionData table.	
53	Input estimate of the percentage of cracked slabs in a segment	The rigid pavement layout of the data entry grid on the testing user interface provides a field for this value to be edited in the RigidInspectionData table.	
54	Display profiler IRI	The testing user interface shows this data after it has been loaded from the WinRP profiler output into the RigidInspectionData table.	
55	Display profiler Ride Number	The testing user interface shows this data after it has been loaded from the WinRP profiler output into the RigidInspectionData table.	
56	Display previous year values for items 41-55	The rigid pavement layout of the data entry grid on the testing user interface provides a field for viewing these values from the RigidInspectionData table.	
57	Display verification flag	All data passes through the validation engine as soon as it is entered into the system.	
WinRP Integration			
58	Automatically run WinRP when a test section is completed.	Currently this requirement is not implemented in the system as designed.	
59	Automatically pull in WinRP data from WinRP text files to the RCI database.	Currently this requirement is not implemented in the system as designed.	
60	Validate collected data with the vehicle database.	The validation engine automatically handles this as soon as data is loaded from the profiler output file.	
61	Identify segments which need to be retested before leaving test section.	The validation engine automatically handles this as soon as data is loaded from the profiler output file.	

Table D1. Requirements compliance matrix (con't)

#	Description	Compliance	Assumptions
62	Commit section test data to the vehicle database.	The validation user interface allows the data to be committed to the proper inspection table.	
63	Show % test sections completed.	The Navigator object automatically tracks this information.	
64	Show % test sections on schedule.	The Navigator object automatically tracks this information.	
65	Show % test sections validated.	The Navigator object automatically tracks this information.	
Quality Control			
66	Check collected data against base quality standards.	The validation engine automatically checks data as soon as it is entered into the XPCS system when possible and practical. If automatic validation is not possible or practical, a user function (i.e., button) will be provided to perform validation checks.	
67	Migrate data quality standards from XPCS database (should be read only for the mobile software)	Validation rules are stored in the ValidationRules table and are synchronized out to the collection database.	
68	Visually flag erroneous data	As soon as data fails the validation checks, the field is visually flagged by the UI.	
69	Synchronize vehicle database with XPCS database	The XPCSSynchronizer object uses a "last wins" rule to logically merge vehicle and central databases.	By "vehicle database" we mean the vehicle's copy of the XPCS database.
GPS Tracking			
70	Automatically track the data collection van's location on a GIS map.	This requirement is handled automatically by the MapPoint library	
71	Provide GPS data for MapPoint.	The Navigator object feeds GPS data into MapPoint.	MapPoint is an SDK we will use to directly embed its functionality within the application.
72	Display the vans location on a map at all times to the user.	The van location is visible at all times while the user is at either the navigation or collection screens assuming a GPS signal is available.	
73	Identify the most efficient route to the desired test location on the GIS map.	This requirement is handled automatically by the MapPoint library	

Table D1. Requirements compliance matrix (con't)

#	Description	Compliance	Assumptions
74	Issue audible commands via text to speech to guide the data collection personnel to the desired test location.	This requirement is handled automatically by the MapPoint library	
75	Show nearest test section on interface	Currently, the routing will take the user to the next test section in the generated sequence (unless they rearrange the sequence).	
76	Show next test section on interface	In the navigation UI, the next test section is always shown.	
Application Logging			
77	Log user usage time	The Navigator object stores this in the RouteAssignments table.	
78	Log system errors	System errors are caught and written to a dump file.	
79	Log all synchronization and validation events	The XPCS synchronizer produces a log of the most recent sync.	

System Description

The mobile system has the responsibility of facilitating the direct data collection process. To that end, it walks the user through the process of logging on, loading the run that's been assigned to them, navigating to test sections, entering survey results for each segment, and validating those entries at the segment and section levels.

System Software Architecture

The Mobile XPCS system is broken down into the several major software modules as shown in Figure D1.

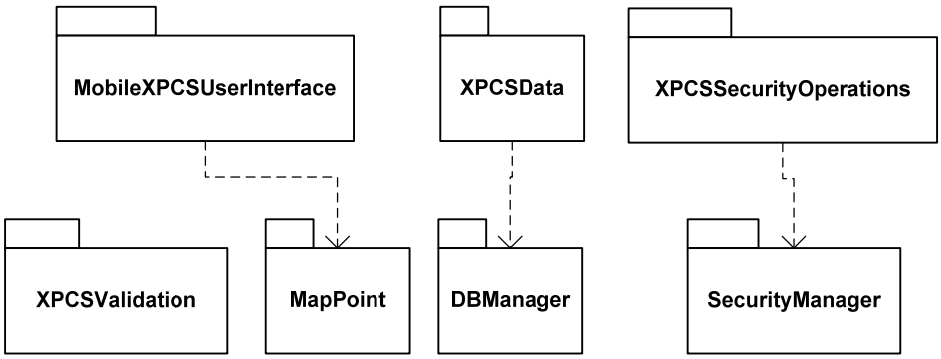

Figure D1. Module Diagram

MobileXPCSUserInterface

The Mobile XPCS User Interface module. This module houses the user interface elements and MapPoint control extensions. It is the program entry point.

MapPoint

Microsoft's GIS software.

XPCSData

Houses the various factory classes that will create business logic and data structures from the database.

DBManager

This contains the generic database wrapper class we use for all database calls and the Microsoft SQLServer-specific implementation of the class that we make use of for this system.

XPCSSecurityOperations

Provides static classes that hold the specific security check functions for the user interface to utilize.

SecurityManager

Generic security system that provides an easy and flexible method of assigning users to roles, grouping actions and specifying permissions at any level of user/usergroup : action/actiongroup. The SecurityManager is further explained later in the report.

XPCSValidation

Provides the business logic objects that perform data validation.

System Hardware Architecture

This section will describe the overall system hardware and communications and their organization.

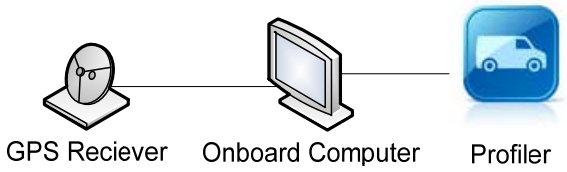

Figure D2. Hardware Architecture

The primary hardware components of the system consist of the following:

1) High Speed profiler
2) GPS receiver
3) Onboard computer
4) USB key

Figure D2 shows the relationship between these components. The USB key is not shown as it is moved by the user between various hardware systems in the vehicle and the office.

External Interfaces

This section will describe interfaces with other software applications or modules.

MapPoint

The Mobile XPCS system will interface with the MapPoint SDK to produce information rich maps which highlight test sections and segments and direct the field user according to the output received from ArcGIS Network Analyst. MapPoint will interface directly with the XPCSData module classes to generate the appropriate information.

Design Approach

This section will detail the order of, approach to, and completion milestones for each of the modules described in the previous section and the approximate amount of time it will take to develop each module and subsystem.

Please note that several modules, once created for the Office XPCS application, will be used in their totality in the Mobile XPCS application, and in the Mobile XPCS Software Design Document are listed with minimal or no development time. These modules include, XPCSData, DBManager, XPCSSecurityOperations, SecurityManager, and XPCSValidation. The XPCS system will reside within a single solution file, allowing us to take advantage of the many duplicate requirements between the two applications. Two separate executables will be generated from the single solution, one for the Mobile XPCS Application and one for the Office XPCS Application.

Total Development Time: 16 days.

Milestone: Completed development of all modules. Able to send application for user testing to FDOT.

Develop MobileXPCSUserInterface

Development Time: 15 days.
Milestone: Completion of all user interface elements as described in this document and the SRD. Successful user testing with FDOT personnel.

The user interface is the key for user interaction with the system. The development of this module will be very iterative with FDOT personnel, allowing them to pinpoint how the interface requirement features should be implemented.

Develop XPCS Database

Development Time: 0 days.
Milestone: Successfully import historical PCS data (already provided by FDOT) into the XPCS database.

Before any of the other modules can really be developed, it will be important to have a working data set for testing purposes in place. This makes development of the XPCS database and the importing of the historical PCS data critical in the design path.

Develop DBManager

Development Time: 1 day.
Milestone: Completion of standard I/O database functions as they pertain to specific FDOT requirements.

The database manager contains classes and functions for storing, retrieving, and updating information contained within the XPCS system. ARA will ensure that the DBManager will function in the Mobile XPCS application as it functions in the Office XPCS application.

Develop XPCSData

Development Time: 0 days.
Milestone: Completion of underlying data access classes and successful testing of DAL through the business logic with prototype user interface elements.

The XPCS Data Module contains all of the relevant data access classes and provides an object-oriented design for the database subsystem.

Technical Specifications

MobileXPCSUserInterface

This module contains all of the form classes the user will interact with while using the mobile application.

XPCSData

This is the factory floor for the majority of the business logic objects in the application. This module

contains the data representation, route, and navigator classes that the user interface will hook events into in order to time the transition between navigation and inspection screens, update the routing screens, and provide access to appropriate Section and Segment objects for the current part of RoutePlan being driven. An illustration of the objects that are members of this module are shown in Figure D3.

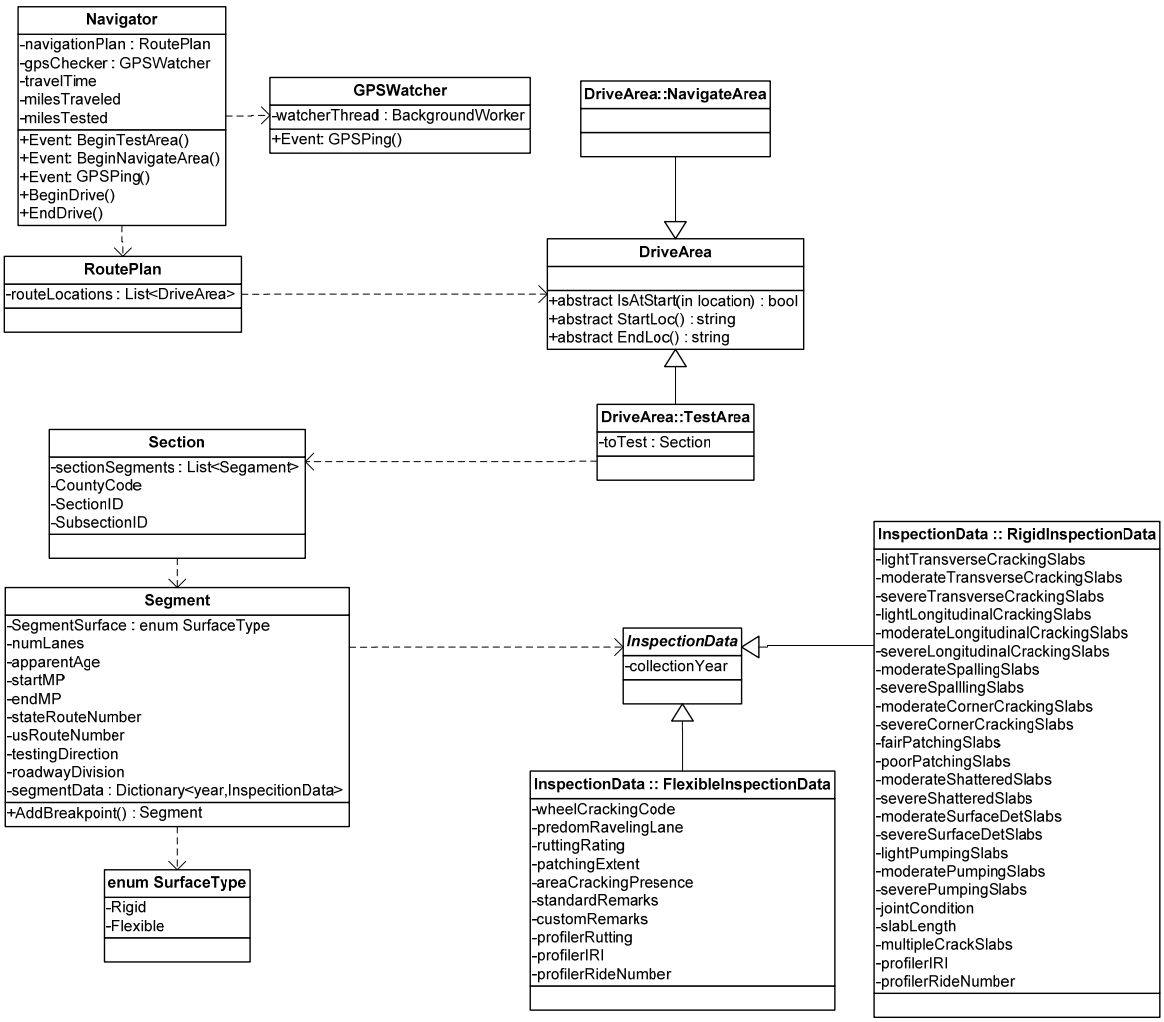

Figure D3. XPCSData module objects

DBManager

This module's primary purpose is to make the abstract Database class (and a SQL Server specific implementation class) available to the rest of the program. This class acts as a wrapper for database calls and creates the connection, command, and adapter objects when queries are run.

XPCSSecurityOperations

This module's primary purpose is to supply a static SecOps object to the user interface. The SecurityManager is a very general system that handles resolving a user specifier/action specifier pair

into some permission level. The purpose of the SecOps object is to hold well named functions that contain the context-specific blueprints of the action specifiers for each relevant security check. This allows for a highly granular breakdown of permissions as needed without cluttering user interface code with large amounts of action specifier code.

SecurityManager

The primary purpose of the module illustrated in Figure D4 is to supply the SecurityManager class to be used by the SecOps layer. SecurityManager is responsible for resolving a descriptor of a specific user and a specific action into a permissions level (None, ReadOnly, ReadWrite, Create, CreateDestroy) against a set of (user/usergroup)->(action/actiongroup) permissions. Permissions are evaluated from least specific to most specific (with user specificity trumping action specificity) in the following way:

1. Usergroup->Actiongroup
2. Usergroup->Action
3. User->Actiongroup
4. User->Action

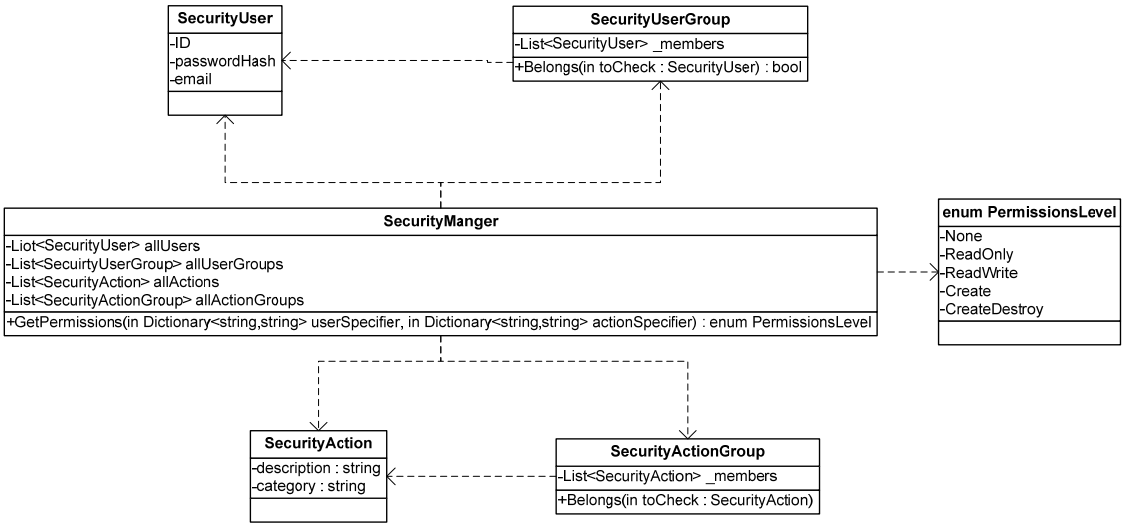

Figure D4. Security manager objects

XPCSValidation

This module supplies the RigidSegmentValidator, RigidSectionValidator, FlexibleSegmentValidator and FlexibleSectionValidator classes. These classes are used to verify the entry of data and flag whether or not the section needs to be retested. These classes implement the base XPCSValidator class and are themselves composed of internal XPCSValidator objects that check specific requirements derived from FDOT's documentation. An example of an existing data validation process currently used by FDOT is shown in Figure D5.

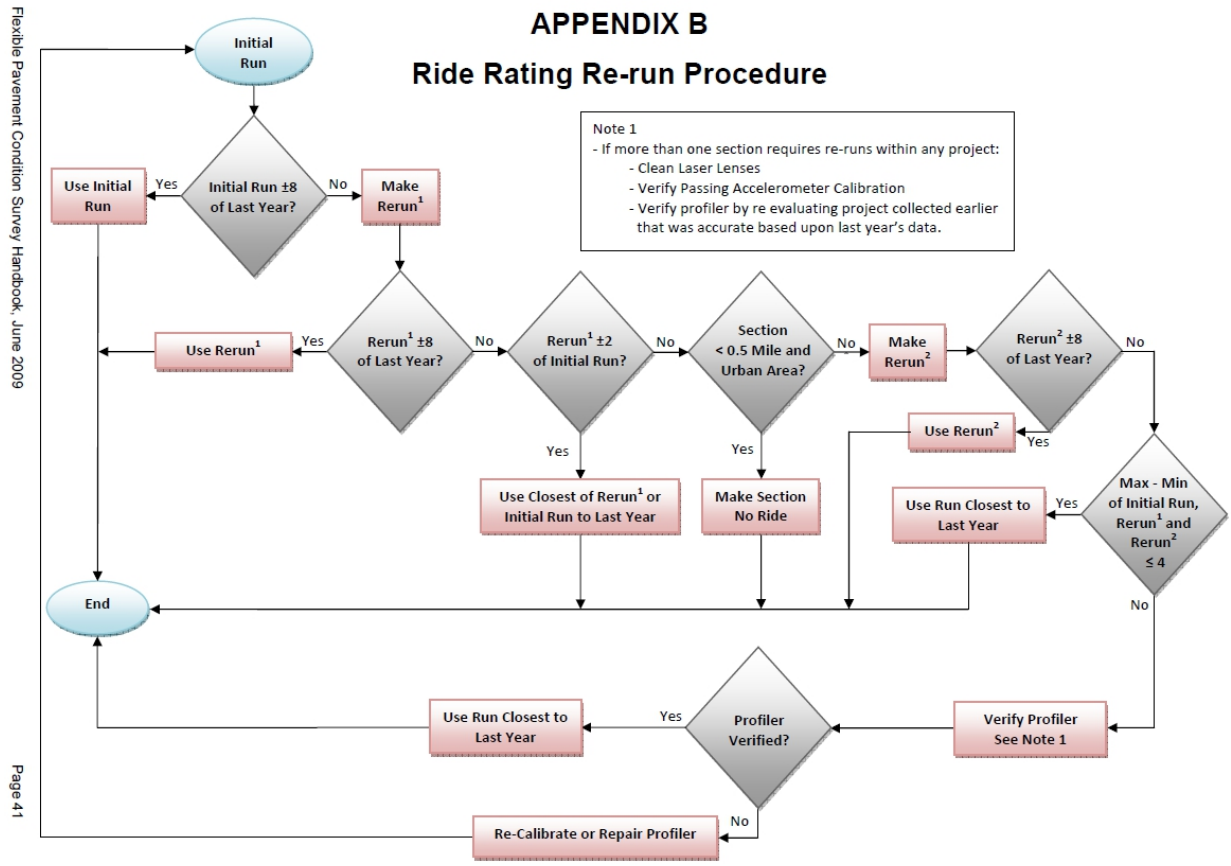

Figure D5. Ride rating rerun procedure

The validation classes all serialize into basic (human readable/editable) descriptors of input and rules (which can then be used by the ValidatorFactory to produce the current rule objects for checking each segment/section. The objects involved in the validation process are shown in Figure D6.

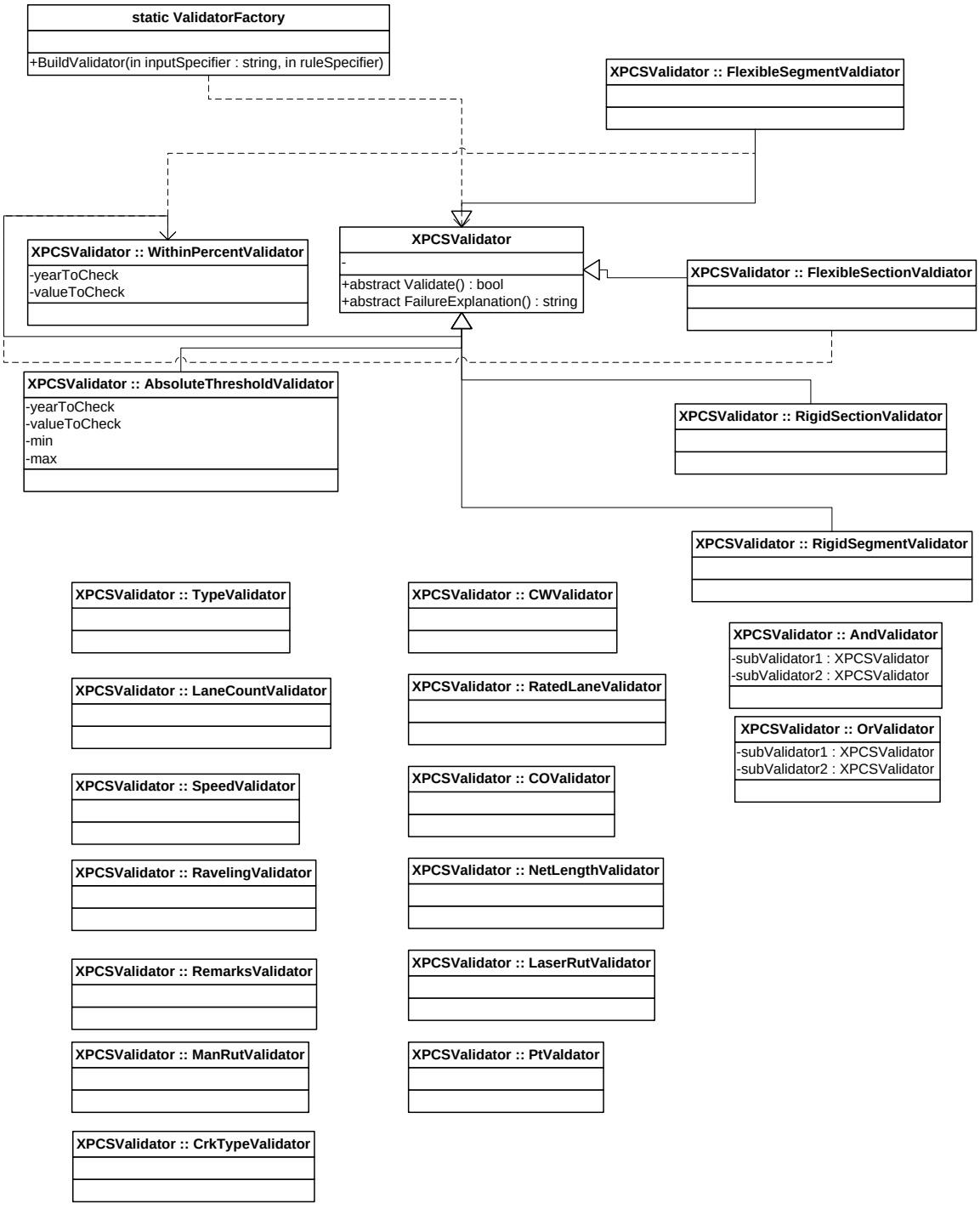

Figure D6. Validation module objects

User Interface Design

The user interface design will be managed dynamically with FDOT staff, within the bounds of the interface requirements as described in the Mobile XPCS SRD.

Data Architecture

This section of the design will discuss the database schema and design decisions made regarding how data is stored in the XPCS system.

Database Management System Architecture

The database schema used to represent the inspection data and security settings is shown in Figure D7. The individual objects within this schema are discussed in detail within this section.

Data Conversion

Currently existing survey data exist in the form of SAS datasheets. FDOT has provided a data dump of all their historical PCS data as excel spreadsheets for inclusion in the XPCS database. ARA will develop a "one time application" export of the data into the XPCS database. No further data conversion is required.

Security

Security in the XPCS system is handled via the SecurityManager module. This module allows for the specification of usergroups and actiongroups. By default, there exists a usergroup called "administrators" and an actiongroup called "All Actions" which is automatically populated with new actions and over which they are given the highest level of access (Create/Destroy). While more groups are possible, FDOT's requirements only call for two levels of (non-administrator) permissions: office and field users. As the relevant security actions are discovered, they will be placed into actiongroups specifically set up for each of these usergroups. The user interface code can then check these permissions via the SecOps object and enable/disable user controls as required in order to present them with only valid options.

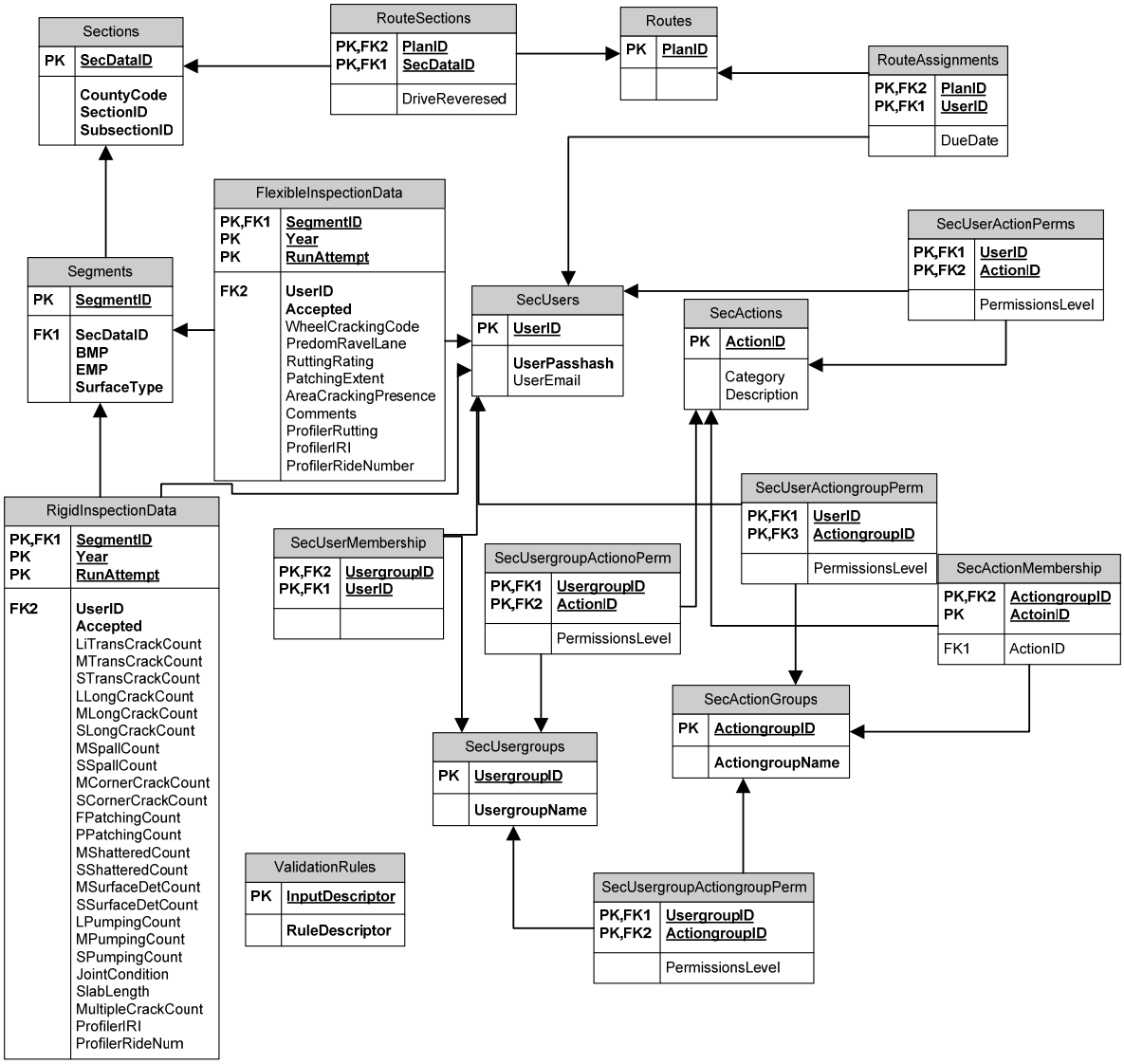

Figure D7. Overall database schema

Capacity

The XPCS system has more pitfalls in terms of complexity in the business logic than in data capacity issues. The amount of data being collected and used is relatively small (in the requirements documentation, 10 GB is the upper limit given and that falls *far* above what is likely to be produced under any likely scenario).

As such, we are free to use what would normally be bulk operations in order to overcome certain technical limitations of the system. For example, the (currently) disconnected nature of the data collection vans necessitates backing up the van database, copying it to a portable device (flash drive, etc.), and restoring it on a database server that can be reached by the XPCS office software in order to begin synchronization (and then reversing the process to get the synchronized data back to the van). If

this system were more data intensive, this process would be onerous.

Performance and Timing

The Mobile XPCS database will be developed with performance in mind. However, performance is not critical to the operation of the application. Due to the small amount of data the application is required to handle, and the performance boost given utilizing an object-oriented "Active Record" design pattern we expect performance to be at better than acceptable levels.

It is much more important that the Mobile XPCS application have increased performance due to the live nature of the application. This is especially importation with regards to the MapPoint Navigation user interface. These performance goals will be realized within the Office XPCS Application due to the close nature of the two programs.

Error Handling

The Mobile XPCS application will utilize C# standard exception handling and will output data dump files to the program directory or users directory when application errors occur. This will help users pinpoint bugs and enable faster turnaround for managing bug resolution. In addition, the users will have access to the Mantis bug reporting system website for the XPCS application and will be able to report bugs and monitor their resolution in a timely manner.

Adaptability

In general, the system's adaptability comes primarily from the parameterization of the validation logic and the extensibility of the security system and the reports. The system should easily expand to further survey vans.

Appendix E

PCS and XPCS Flowcharts

Table of Contents

List of Figures

Description

As a part of the improvement process, ARA and FDOT started with the existing workflow of the current Pavement Condition Survey (PCS) process to determine how to what features and routines should be included in the Extended Pavement Condition Survey (XPCS). This appendix contains a series of flowcharts that show the original process and the process as modified by the XPCS system. Please note that the original process flowchart (Figure E1 through Figure E6) is based on an Excel spreadsheet created by the State Materials Office (SMO). Another item that may be compared between the original and modified process charts are the step numbers; the numbers from the original process were maintained which means that processes that are no longer required have created skips in the numbering within the enhanced workflow. This is intentional to maintain a link between the two process charts.

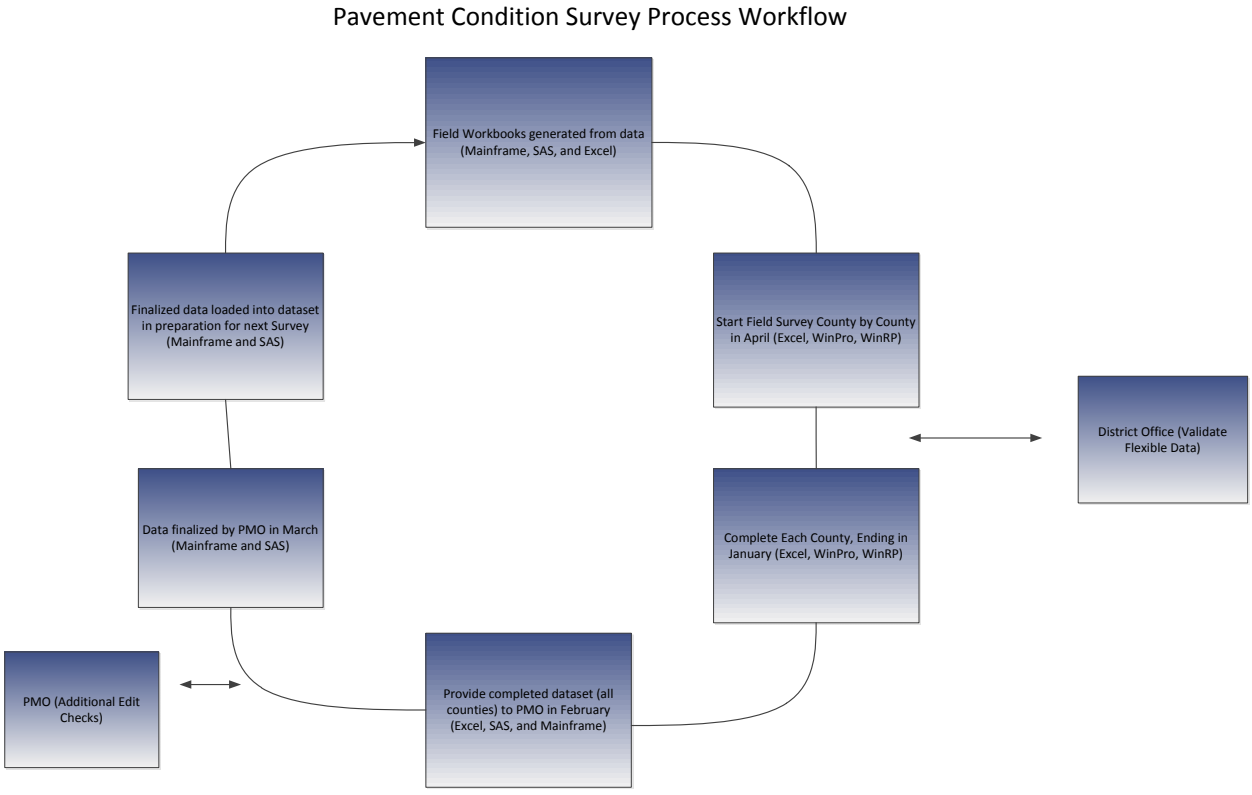

Figure E1. Overview of PCS process

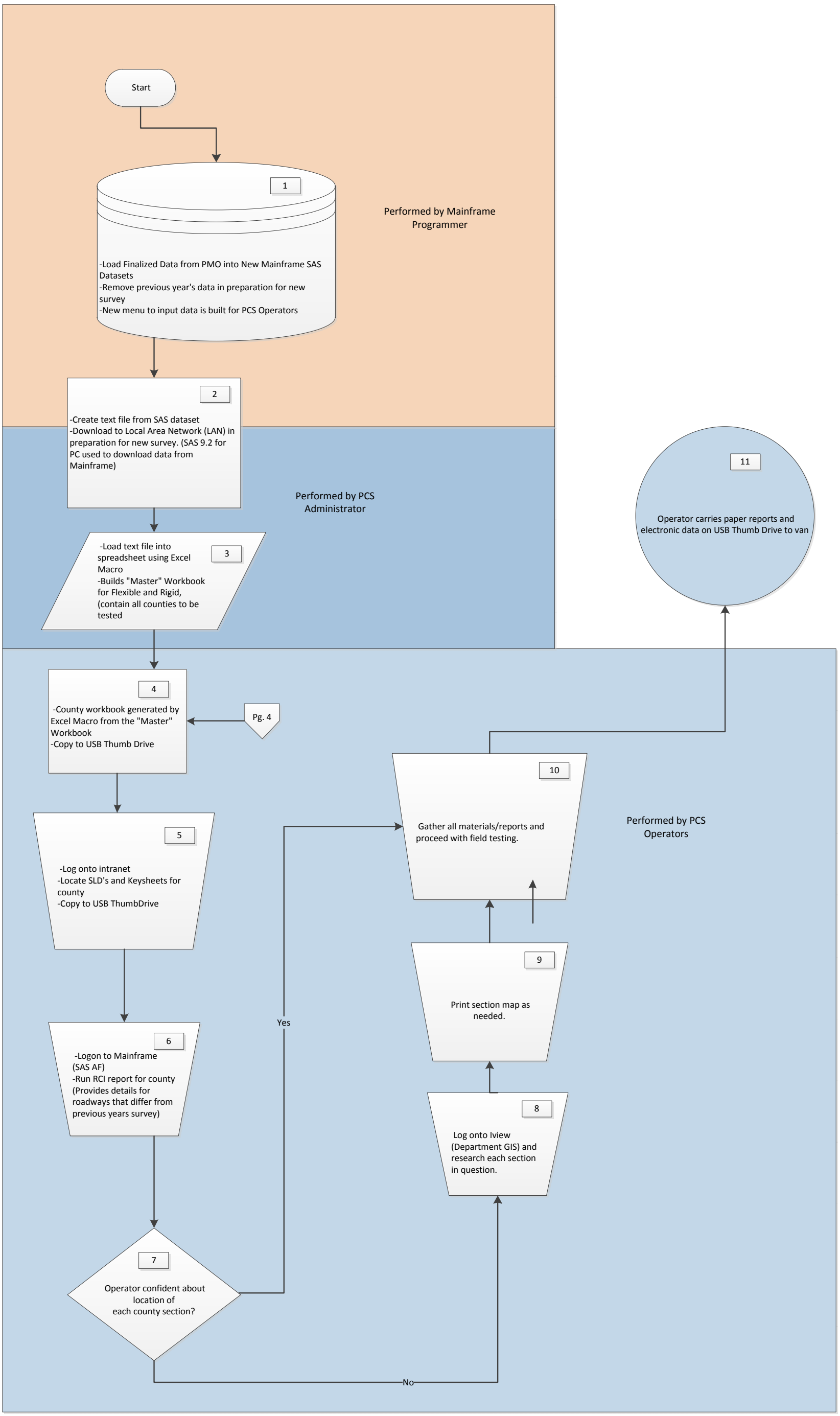

Figure E2. PCS workflow, page 1

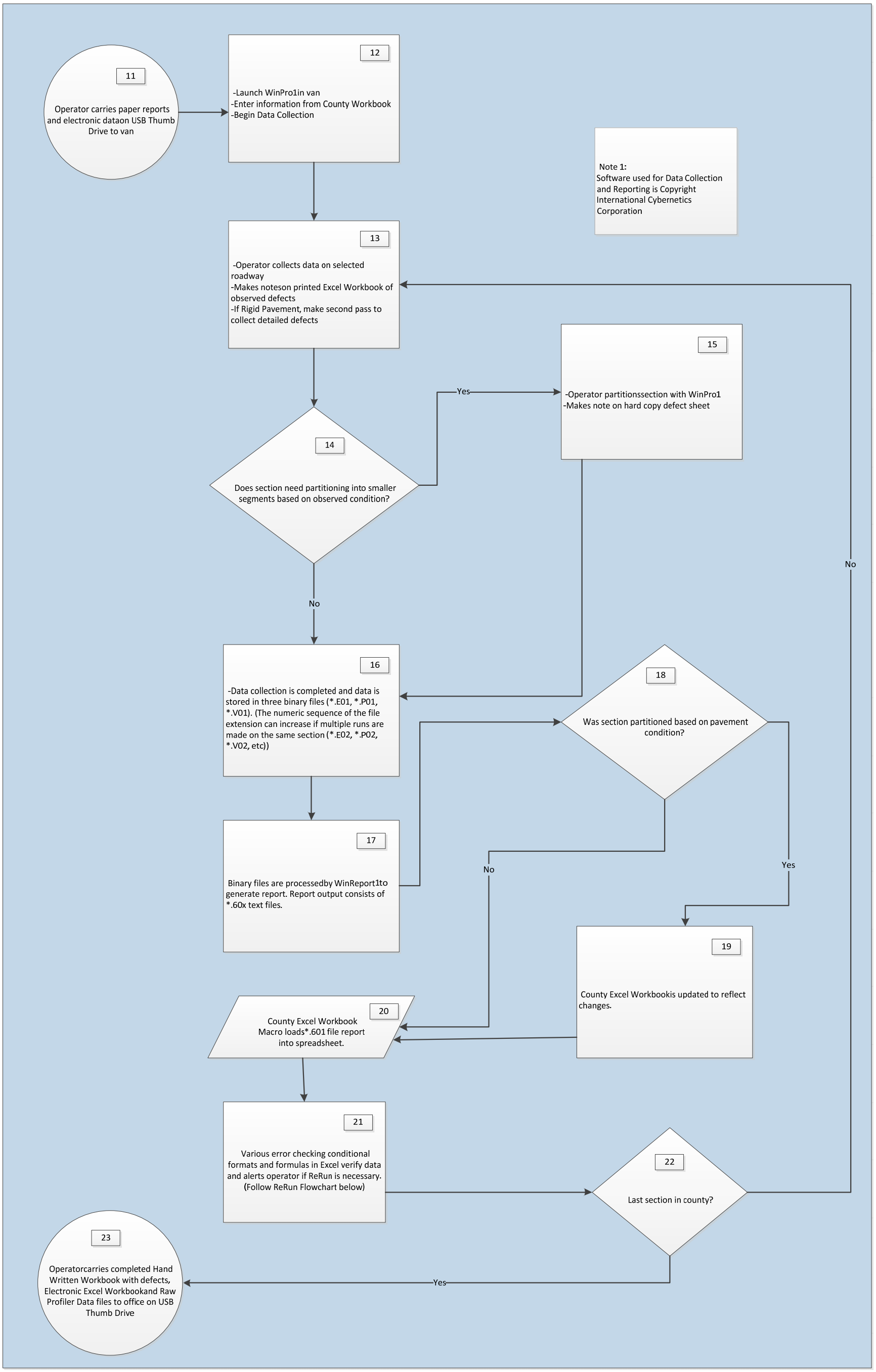

Figure E3. PCS workflow, page 2

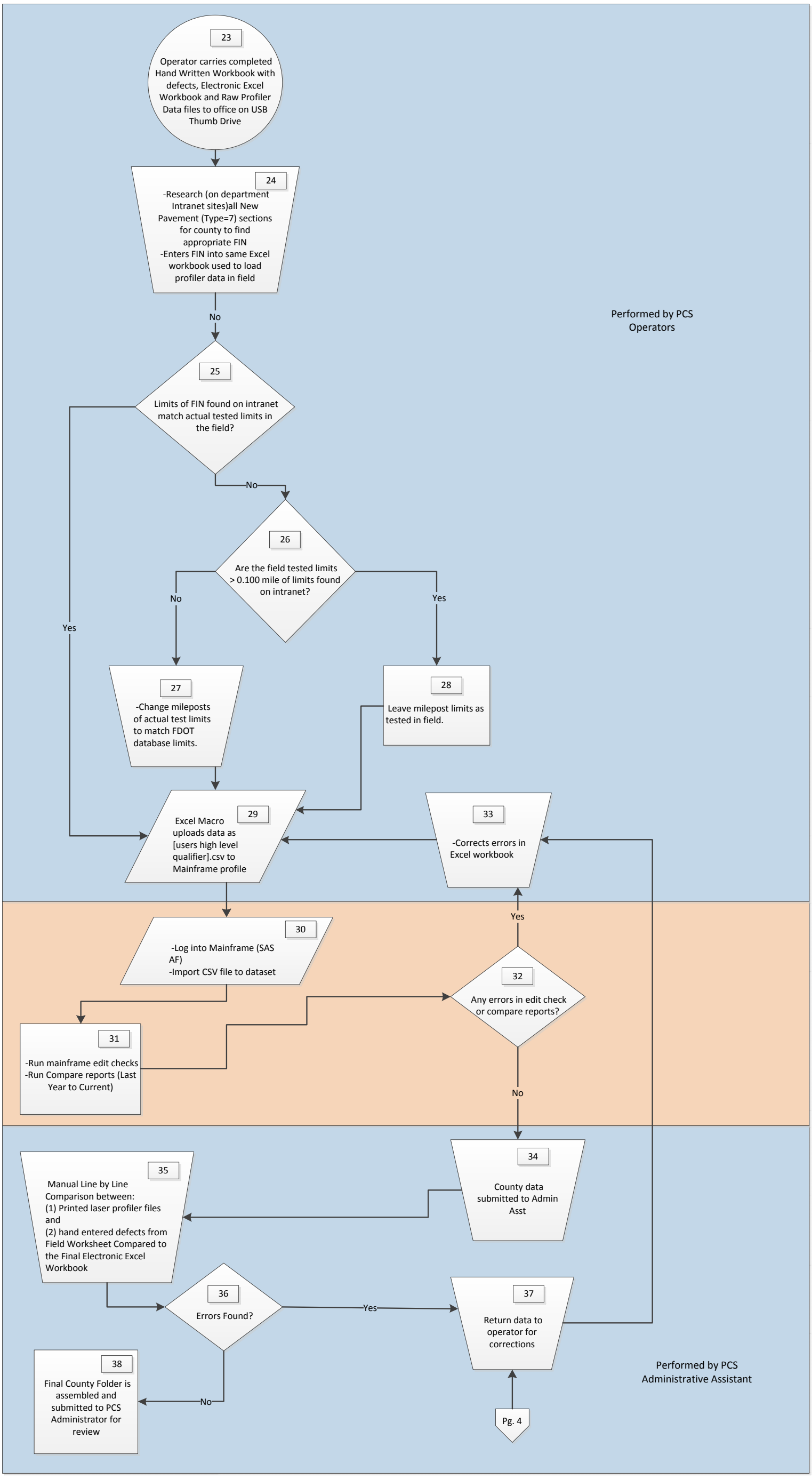

Figure E4. PCS workflow, page 3

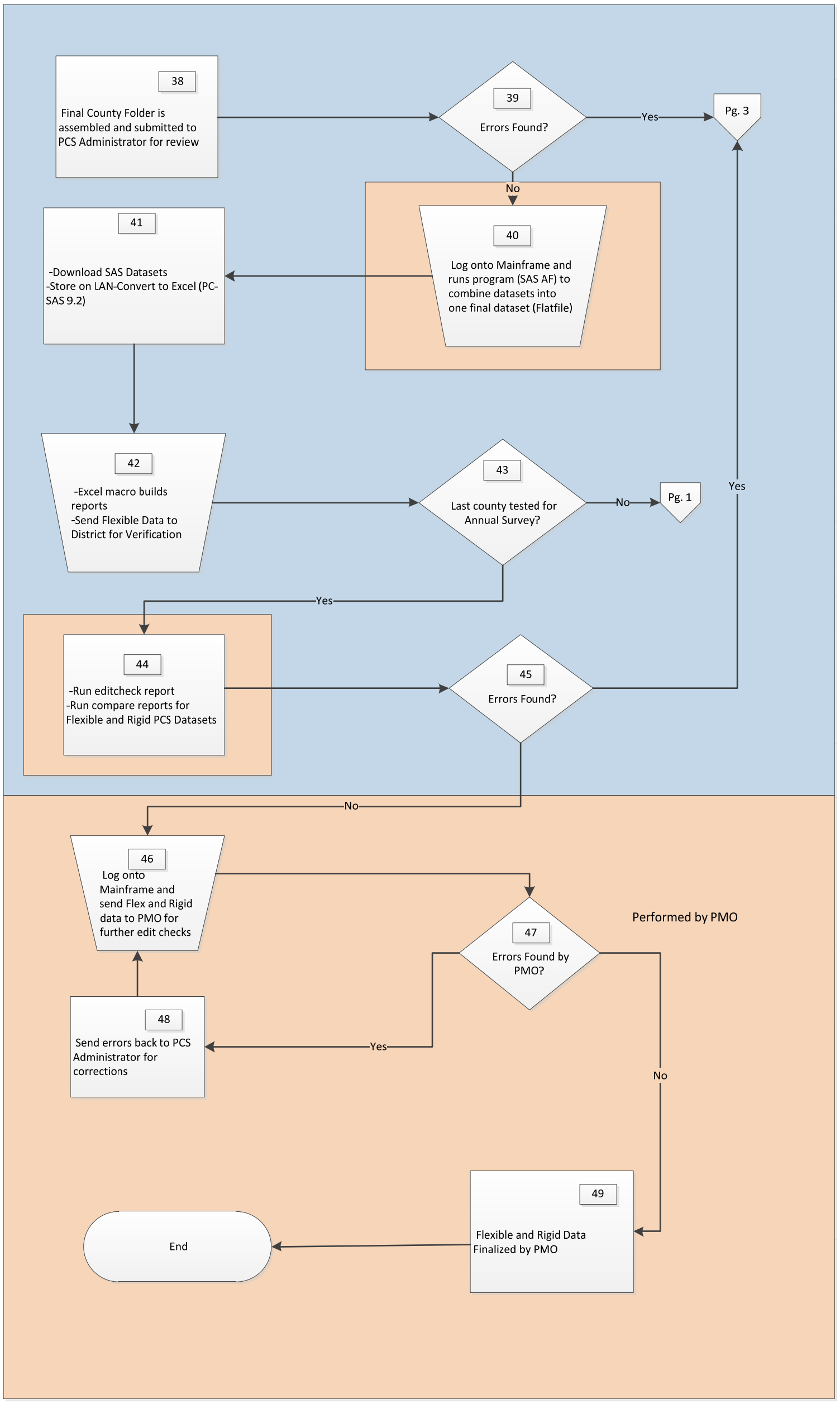

Figure E5. PCS workflow, page 4

Legend & Acronyms

PCS= Pavement Condition Survey
PMO= Pavement Management Office
RCI= Roadway Characteristics Inventory
SLD= Straight Line Diagram (Per Roadway ID)
Keysheet= County Map containing Roadway ID's
FIN= Financial Identification Number (aka, Project Number)
Final County Folder= Contains all reports from Mainframe and Excel Workbooks

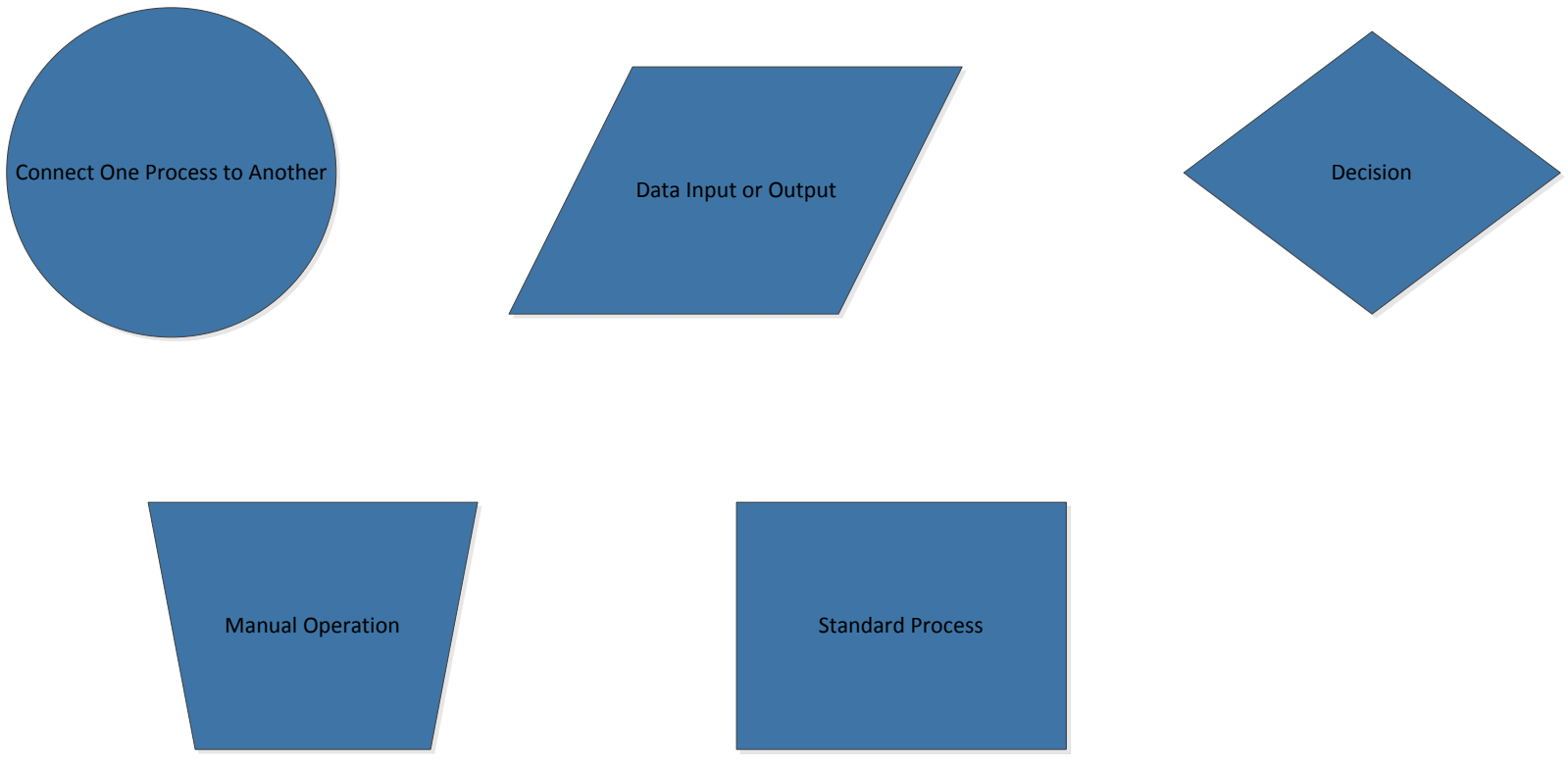

Figure E6. Legend and acronyms used in workflow charts

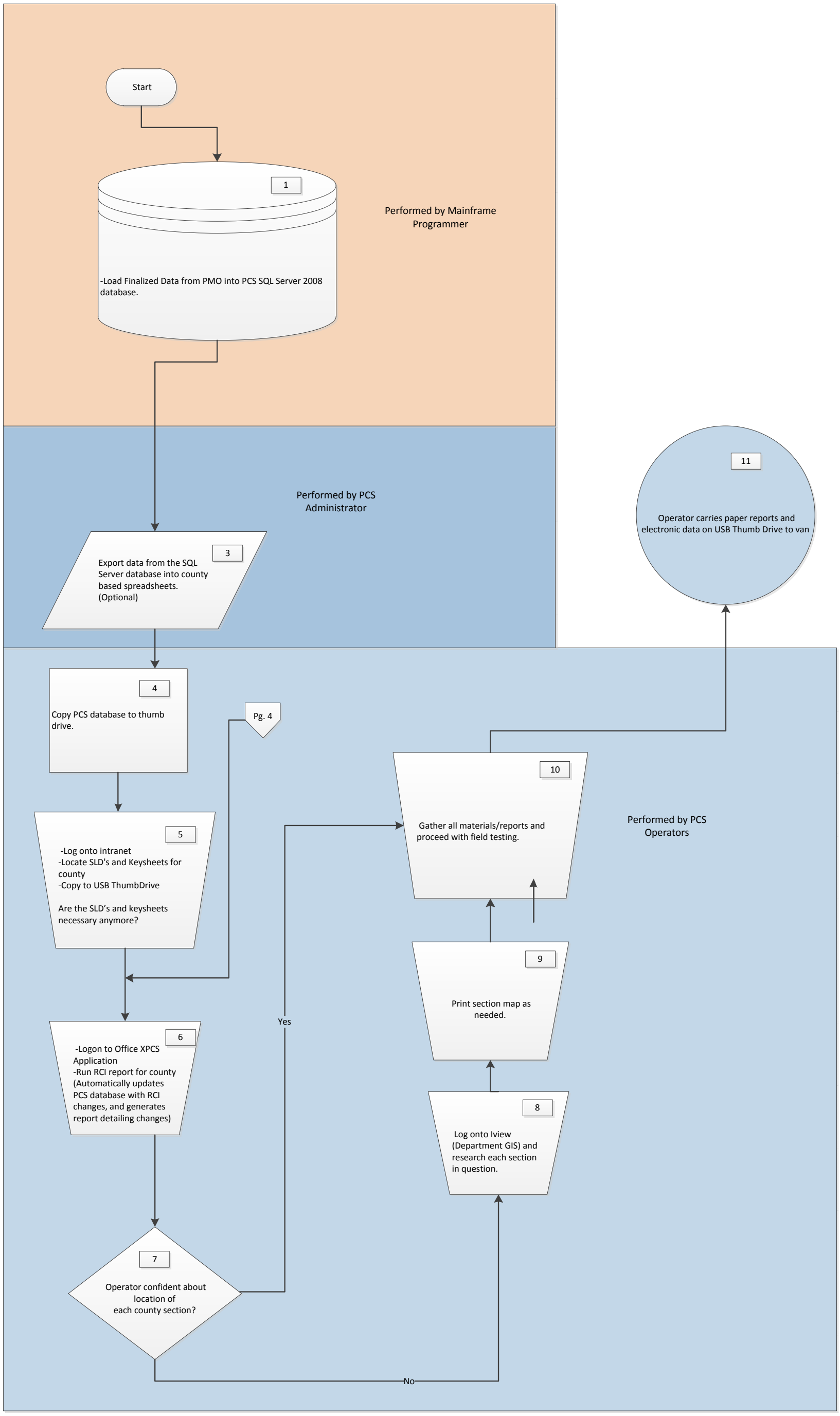

Figure E7. Enhanced workflow, page 1

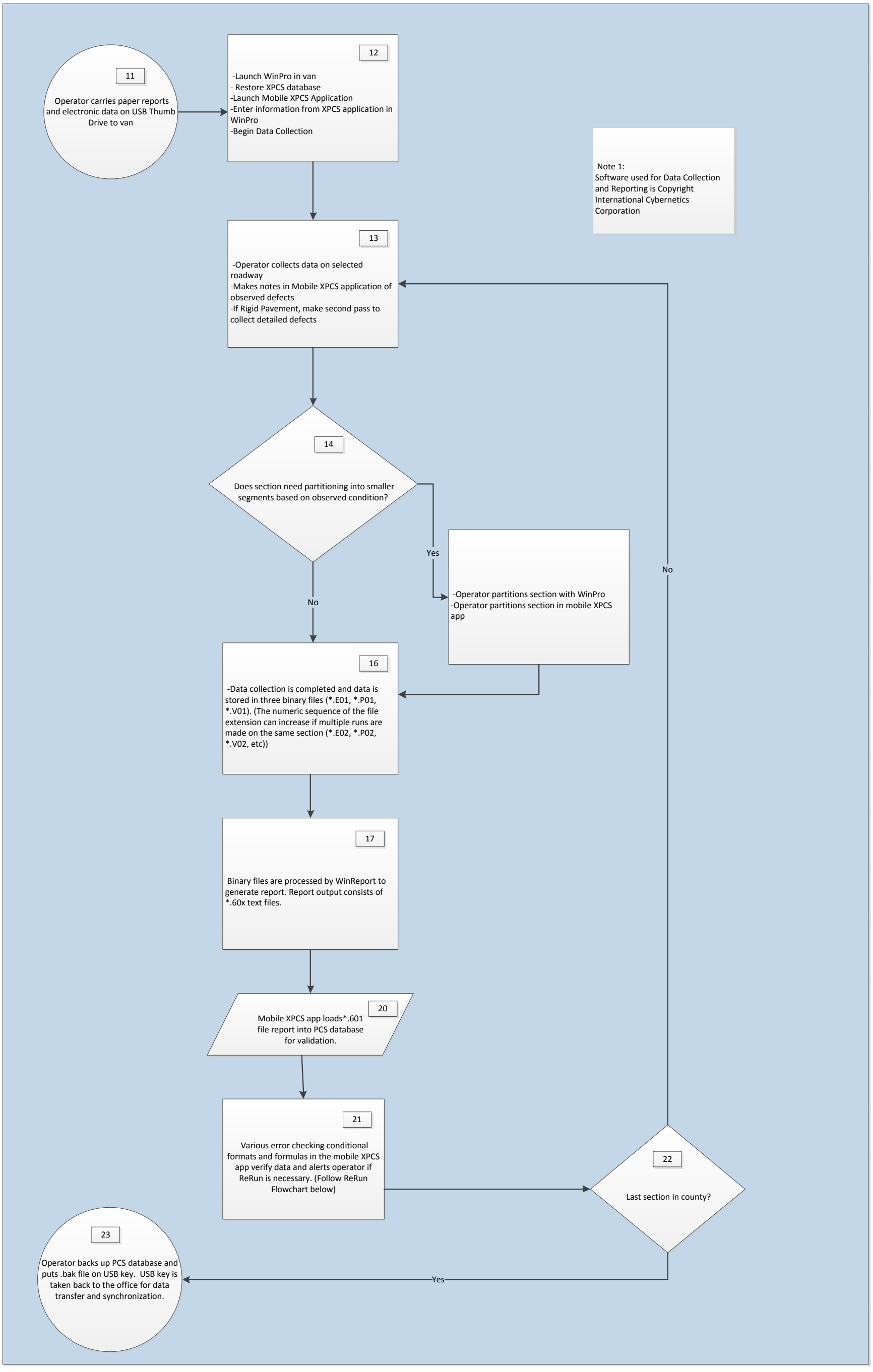

Figure E8. Enhanced workflow, page 2

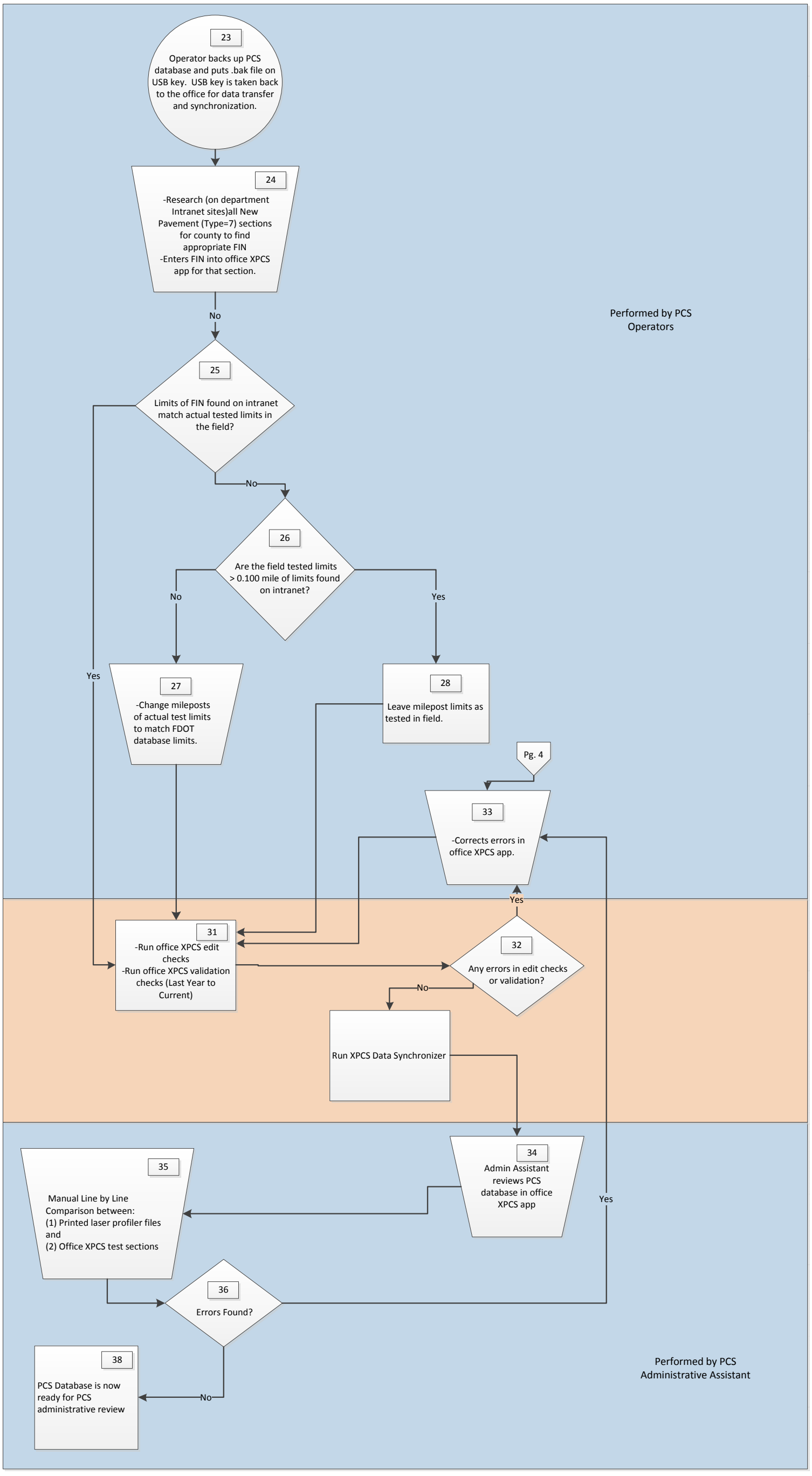

Figure E9. Enhanced workflow, page 3

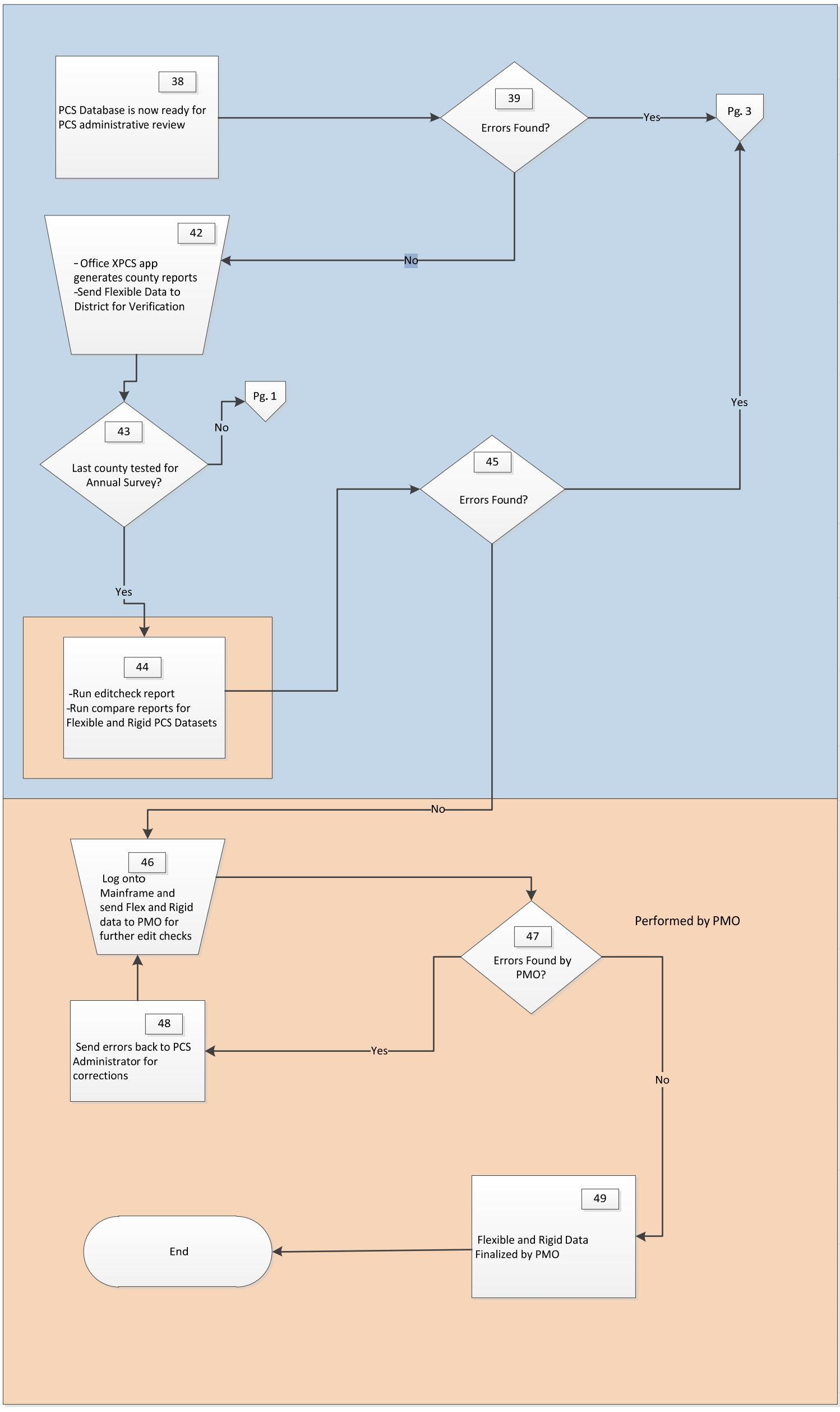

Figure E10. Enhanced workflow, page 4