**Software System Specification and Project Plan**

Comp 450, Software Engineering

**PPG Invoice Finder**

**KRSS**

Team leader: Stephen Tam

Ronald Jenkins, Seth Loew, Kory Stansbury

**Editor of this Document:** Stephen Tam

Date of Delivery: 02/23/2015

PPG

400 Bertha Lamme Dr

Cranberry Township, PA

<http://corporate.ppg.com/Home.aspx>

Contact: Akos “Kosher” Nagy

400 Bertha Lamme Dr

Kosher.nagy@ppg.com

724-742-5246

**1 Narrative description of project**

Invoice Finder will be an application that is set up to search for invoices and display them in various ways. It will work with PPG’s current system by searching in their archive folder, root folder, and caches, looking for user requested invoices. The search is dependent on the search parameters that the user inputs. Once found, those invoices will be displayed in either a .zip folder or a .pdf file. This software will be able to show if an invoice was not found. If an invoice was not found, it is either missing or corrupted. The user will be able to see which invoices were not found and be able to put them into their final destination or request re-generation of the invoice. This saves PPG’s IT department time from searching through the folders manually to find the invoices, increasing the department’s efficiency and the company’s.

**1.1 Context**

Because of PPG’s problem with the periodic loss or corruption of invoices, IT and helpdesk workers spend hours tracking down invoice PDFs that would otherwise be irretrievable. Invoice Finder will operate on PPG employee’s laptops, running Windows 7. The application will access folders on the PPG network that the laptops have access to. Our Invoice Recovery System will quickly search through potentially millions of PDF files, in order to partially automate the cleaning and maintaining PPG’s invoice file management system. PPG in a company reports stated, “We are in the business to serve customers” (PPG, 3). This is one of PPGs core values. Because customers often lose and request old invoices, Invoice Finder will assist to catering to customer’s needs and desires efficiently and promptly.

**1.2 Overview of software product capabilities**

PPG Invoice Finder will provide PPG employees with a simple to use Windows Form application that locates customer invoices based on various search criteria, such as: customer ID, store ID, region ID, invoice number, date range, or any combination of these. PPG Invoice Finder will include multiple user adjustable settings that will extend the usefulness of the application. For example, not only will users be able to adjust locations that the Invoice Finder searches, but users will also be able to adjust the number of locations PPG Invoice Finder searches. Another setting provided will be the ability to choose between exporting the discovered invoice(s) as a ZIP file or large PDF file.

If PPG Invoice Finder determines the location(s) of the requested invoice(s), then it will bundle the discovered invoices into either a ZIP file or a large PDF file, depending on the user’s defined preference, so that the invoices can be easily distributed out to concerned parties. If PPG Invoice Finder cannot find the requested invoices, then it will determine where the invoice was corrupted, so that the user can request a regeneration of the invoice or move the file manually to its final destination.

PPG Invoice Finder will have 3 screens. The first screen will be the search query screen that users input search criteria. The second screen will be the results screen that displays in a table format each invoice, its location, and a select box for exporting. The last screen will be the settings screen.

Because PPG Invoice Finder is only “reading” and copying data, we are not concerned with data corruption caused by it.

**1.3 Related systems.**

Include here at least two examples of software systems which are similar or related to the proposed system. For each example, make explicit reference to at least one published work which describes some aspect of the example system or the environment into which it is deployed.

**2 External view**

Invoice Finder will run inside PPG’s current system. It will communicate with the servers inside of PPG’s network including the Archive folder, the Root folder, and various caches. There will be one interface that the user deals with. The user will be able to search for invoices using different parameters such as customer ID, date, range of dates, and invoice number. Once the user has entered in the specified information, the application will look through the current folders and caches. Once found, the interface will then display invoices from the search results and give the option to export the results as either a .zip folder or a .pdf file of all the invoices. It will also list the invoices that were not found/corrupted.

**2.1 User scenarios**

**Scenario A.**

In order to properly configure our application to perform invoice searches, a PPG IT employee would edit the paths to the Root and Archive folders containing the invoices. Such a behavior is shown below in Use-Case i. This tool, to be implemented as a pop-out file search window, would store the file paths which the user wished to be searched. These file paths would be orderable by priority, and the use would be able to add additional paths to the base two if they so desired.

**Scenario B.**

Proper searches using our application would require the entering of several search parameters for the tool to use as shown in Use-Case ii. In particular, the PPG Help-desk employee using the software could narrow his search by entering a Customer ID number, an Invoice ID number, or date ranges for the search tool to use. In order to simplify the process of locating multiple date ranges worth of invoices, our application would allow the user to enter multiple date ranges at once.

**Scenario C.**

Our application is designed to output the located invoices in one of two ways. A PPG Help-desk employee may have the application output the located invoices in a .zip file, and/or concatenated in a .pdf file. Use-Case iii demonstrates this interaction. In addition to choose which of these outputs the user would like (either or both), the user may also choose whether or not to generate a failure report for the search. This report would indicate any invoices which the program would have expected but couldn’t find.

**Scenario D.**

In addition to setting the Archive and Root folders for the search component of the application, a PPG Help-desk employee would be required to set the output folder for any files creates by the search (Use-Case iv). This would simply take the form of Pop-out file explorer window for locating or creating a destination folder.

**Scenario E.**

Once configured, the program would be ready to search for invoices. A PPG Help-desk employee, as shown in Use-Case v, would request a search from the system. Our system would then draw from the specified Archive and Root folders and perform a search based on the parameters that the user had provided. Finally, output files of the invoices would be generated and placed in the specified destination folder.

**2.2 Use-case diagrams**

Each of the scenarios in paragraph 2.1 should appear in, and refer to, one of a series of ucd’s, produced in Visio and pasted into the document at this point. Label the ucd’s with lower-case Roman numerals (i, ii, iii, etc). Each of these should be referenced at least once in paragraph 2.1.

**2.3 Integration issues**

There are several issues when integrating our application with PPG’s current system. Security issues limit the amount of testing that can be done at this time. The Invoice Finder should look through different invoices and once those invoices are found, the application will display those invoices. Due to security issues, we are not able to search the invoices of PPG’s customers, we must make our own “customers” and make them as similar to PPG’s invoices as possible. This could potentially cause problems when integrating our application. We must be careful with how we program our application to make sure that it is easy to understand and interpret in order that PPG IT employees are able to change the necessary code. That way, PPG can use our application with their system. Failure to do so could result in the application not working with PPG’s current system even though the application works with our “customer” invoices.

**3 Breakdown of software components to be produced**

The proposed system consists of 3 major components, each having an average of 3 subcomponents.

**3.1 Component 1 – User Interface**

Invoice Finder includes a Graphical User Interface that the user will interact with. The main screen will give the user different parameters to search the invoice folders.

**3.1.1 Search Parameters subcomponent**

The user will be given parameters to search for invoices. Each invoice has a specific identification number including the date of the transaction, the customer number, and the invoice number. This data will then be used to query the database for a full filename or will be used to search files in the set folders. The user may enter more than one parameter when searching for an invoice. The application would look at every parameter and make its search based off of those parameters.

**3.1.2 Results subcomponent**

The user will interact and filter through the list of invoices that the Invoice Finder recognized as matches according to the search criteria. The Results page will consist of a table view of the invoices.

**3.1.3 Settings subcomponent**

The user will be able to adjust which folders the Invoice Finder will search through and also adjust how many attributes the results page displays. The settings page will

**3.2 Component 2 – Search/Process Invoices**

Once the user enters the search parameters, the application will process those parameters into a query. The query will include the different parameters that the user entered and search the folders and the caches to find the invoices. Then it will send those invoices back to the Invoice Finder.

**3.2.1 Retrieve data from the search parameters subcomponent**

The data that the user entered will be received into a query. The query will consist of the parameters that the user has entered.

**3.2.2 Search for the invoice subcomponent**

The query will search into PPG’s system to find the specific invoices. The search will look into the archive and root folders. It will also look into the various caches where the invoices could be.

**3.2.1.1 Priority queue subcomponent**

There will be a priority queue on the order of where the query will look first. The query will first look into the caches, then it will look into the root folder, and the archive folder.

**3.3 Component 3 – Output File Manager**

After the search for the invoices is finished, the application will display all of the found invoices in the Results subcomponent of the user interface. From there users will be able to export the found invoices as a .zip or .pdf file and a document listing the invoices that were missing/corrupted.

**3.3.1 .zip subcomponent**

One of the ways the invoices will be shown is by a .zip folder. In the folder will be the list of invoices that were found with the invoice number showing.

**3.3.2 .pdf subcomponent**

The other way that the invoices will be displayed is by a .pdf file. This .pdf file will be a large file with all of the invoices in that specific .pdf file.

**3.3.3 Corrupted/Missing Invoices subcomponent**

While the query is searching for the invoices, if an invoice could not be found, the application will keep track of that. When the query is done searching for the invoices, it will have a list of invoices that were not found. These invoices will be produced in a list that will be displayed for the user. The user knows which invoices were not found and will be able to put the invoices that were not found to their final destination or send out a document with the missing invoice names, so that a request for re-generation can be made.

**4 Function Point Analysis**

Team KRSS estimates the project encompasses 65 function points. We estimate the total effort at 135+XX person-person hours, for an average of 2 person-hours per function point.

**4.1 External Inputs (EIs)**

The project incorporates 3 EIs, with a weighting factor of 5 (5, simple, 7 average, 10 complex), for a total count of 15.

**4.1.1 Searching for Invoices**

This function uses control inputs from the user to determine which invoices to return to the user in the Invoice Search Results ILF. This function will use textboxes and pick lists to record inputs and then format the inputs so that the program can search the folders for matches. This has a simple weighting factor because it only draws input from the user.

**4.1.2 Change Results Filter**

The results returned to the user after the initial search can be sorted and filtered and this function will utilize textboxes and pick lists to manage control inputs. The Filtered Search Results ILF will be adjusted according to the values returned by this function. This task has a simple weighting factor.

**4.1.3 Change Settings**

The PPG Invoice Finder will allow the user to change the folders the application searches through for invoices, so this function will modify the Search Locations ILF. The user will use textboxes and windows pop up browser to modify the folders. This EI has a simple weighting factor.

**4.2 External Outputs (EOs)**

The project incorporates 3 EOs, with a weighting factor of 7 (5 simple, 7 average, 10 complex) for a total count of 21.

**4.2.1 Sort Results**

The search results will be capable of being sorted. The sorts include rearranging by customer ID, store ID, region ID, or date. This will be accomplished by a series of value comparisons. This function will be automatically called once when the invoice search results are returned, sorting the data by date (newest to oldest). This task has an average weight factor.

**4.2.2 Find Invoices**

When searching for invoices matching user criteria the software will perform a series of comparison statements to determine matching invoices. Because the user can give varying specificity, the number of comparisons resulting as matches (true) could be very large, resulting in a lot of processing. This function will be used every time the user does an invoice search. This task has a complex weighting factor due to the response time hurdle that will result from the thousands of invoices that could end being compared processed.

**4.2.3 Filter Results**

This function will use a simple comparison statement to filter out invoices in the search results. This function will be very similar to the Find Invoices EO. This will be utilized every time the user enters filter criteria and hits the filter button. This task has a simple weighting factor.

**4.3 External Inquiries (EQs)**

The project incorporates 0 EQs.

**4.4 Internal logical files (ILFs)**

The project incorporates 3 ILFs, with a weighting factor of 10 (7 simple, 10 average, 15 complex), for a total count of 30.

**4.4.1 Invoice Search Results**

This ILF is a table consisting of the invoice file names that matched based on the search criteria, their location, transaction data, an email request box, and an export select box. The data displayed in this table can be adjusted in the settings. This task has an average weighting factor due to the learning curve involved with abstracting the GUI from the C# code.

**4.4.2 Filtered Search Results**

The set of invoices returned to the user based on the search criteria can be further narrowed down and sorted with filter settings. The filter options will be the same options available on the search criteria page. The data can be sorted by value based on the various data associated with the invoice file name. This task has an average weighting factor due to the learning curve involved with abstracting the GUI from the C# code.

**4.4.3 Search Locations**

The folders that the application searches in for invoices are displayed in a basic table. The table displays the folders path, its internal nickname, and its status upon application startup. This task has an average weighting factor due to the learning curve involved with abstracting the GUI from the C# code.

**4.5 External Interface Files**

The project incorporates 0 EIFs.

**4.6 Value Adjustment Factors (VAFs)**

The sum of all VAFs is 34, based on the following weights (0 to 5, where 0 indicates the factor has no relevance and 5 indicates that the factor is absolutely essential).

|  |  |
| --- | --- |
| Factor | Weight |
| 1. Backup and recovery | 0 |
| 1. Data communications | 2 |
| 1. Distributed processing | 0 |
| 1. Performance | 5 |
| 1. Heavy use | 5 |
| 1. On-line data entry | 0 |
| 1. Multiple screens | 2 |
| 1. On-line update | 0 |
| 1. Data complexity | 2 |
| 1. Processing complexity | 3 |
| 1. Reusability | 5 |
| 1. Conversion/installation | 5 |
| 1. Multiple installation | 5 |
| 1. Facilitate change | 0 |
| WEIGHTED SUM | *34* |

**4.7 Function point computation**



The weighted sum of all information domain counts (from sections 4.1-4.5) is 66, and the weighted sum of all VAFs is 34, giving a function point count of:

FP = 66 \* [0.65 + (0.01 \* 34)] = 65.34.

This rounds down to a final function point count of 65.

**5 Task breakdown for the conduct of the project**

Using a task breakdown analysis, team KRSS has identified 26 individually scheduled subtasks, requiring a total of XX person-hours.

**5.1 Documentation preparation**

9 separate documents must be prepared, requiring a total effort of 42+XX person-hours.

**5.1.1 Subtask—Project description and elevator talk**

This task involved 4 individuals and required 4 person-hours

**5.1.2 Subtask—System specification / Project plan and classroom presentation**

This task involved 4 individuals and required XX person-hours

**5.1.3 Subtask—Logic Flow diagram**

This task will involve 4 individuals and require 4 person-hours.

**5.1.4 Subtask—Test-Case list**

This task will involve 4 individuals and require 2 person-hours.

**5.1.5 Subtask—Increment 1 plan**

This task will involve 4 individuals and require 4 person-hours.

**5.1.6 Subtask—Increment 2 plan**

This task will involve 4 individuals and require 4 person-hours.

**5.1.7 Subtask—Midterm Report**

This task will involve 4 individuals and require 8 person-hours.

**5.1.8 Subtask—Increment 3 plan**

This task will involve 4 individuals and require 4 person-hours.

**5.1.9 Subtask—Final Report**

This task will involve 4 individuals and require 12 person-hours.

**5.2 Program objects and functions construction**

7 separate deliverables must be created, requiring a total effort of 32 person-hours.

**5.2.1 Subtask—Create test data and environment**

It will involve 2 individuals and require 4 person-hours.

**5.2.2 Subtask—Build logic flow diagram of data within program**

This task will involve 4 individuals and require 6 person-hours.

**5.2.3 Subtask—Construct Invoice object**

This task will involve 2 individuals and require 6 person-hours.

**5.2.4 Subtask—Construct Finder object**

This task will involve 2 individuals and require 6 person-hours.

**5.2.5 Subtask—Construct Settings object**

This task will involve 2 individuals and require 4 person-hours.

**5.2.6 Subtask—Construct Results object**

This task will involve 2 individuals and require 3 person-hours.

**5.2.7 Subtask—Construct Exporter object**

This task will involve 2 individuals and require 3 person-hours.

**5.3 Graphical user interface construction**

5 different GUI tasks must be completed, requiring a total effort of 31 person-hours.

**5.3.1 Subtask—Construct interface allowing input of search criteria**

This task will involve 4 individuals and require 6 person-hours.

**5.3.2 Subtask—Construct interface for adjusting search and export settings**

This task will involve 4 individuals and require 10 person-hours.

**5.3.3 Subtask—Construct interface to view and operate on results**

This task will involve 4 individuals and require 8 person-hours.

**5.3.4 Subtask—Construct interface for filtering and sorting results**

This task will involve 2 individuals and require 4 person-hours.

**5.3.5 Subtask—Link the graphical user interface with the program**

This task will involve 2 individuals and require 3 person-hours.

**5.4 Testing and confirm database compatibility**

4 subtasks must be completed, requiring a total effort of 22 person-hours.

**5.4.1 Create more use and test cases**

This task will involve 4 individuals and require 8 person-hours.

**5.4.2 Create Test Database**

This task will involve 2 individuals and require 2 person-hours.

**5.4.3 Create Queries to retrieve full filenames**

This task will involve 2 individuals and require 2 person-hours.

**5.4.4 Modify search functions and objects to accept query results**

This task will involve 4 individuals and require 10 person-hours.

**5.5 Deploy and train PPG users**

1 subtask must be completed, requiring a total effort of 8 person-hours.

**5.5.1 Present final product to PPG, deploy the product, and train users**

This task will involve 4 individuals and require 8 person-hours.

**6 Schedule**

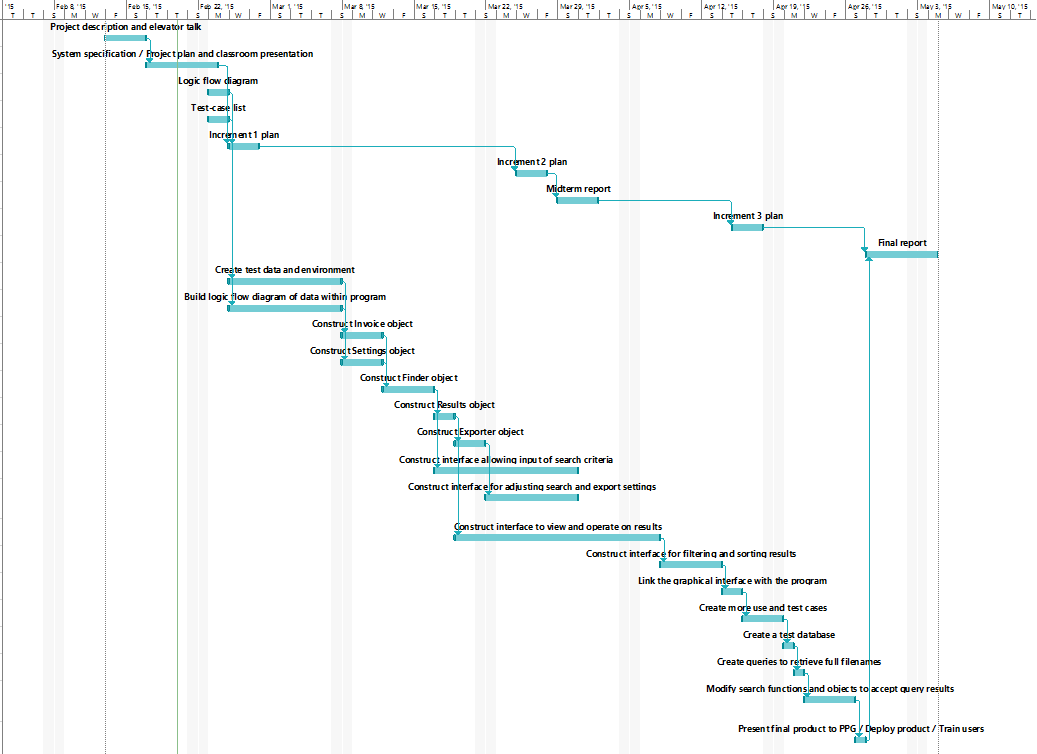
The schedule covers an elapsed time of 80 days, and includes 26 individually scheduled subtasks.

KRSS recognizes that the estimates made could deviate significantly from the actual timeline of completed tasks. The article “Disruption Management for resource-Constrained Project Scheduling” offers several solutions for project recovery, when the project gets off the planned schedule. A couple that will be utilized by KRSS are: rescheduling, resource alternative, and activity cancellation. (Zhu, Bard and Yu, 368) Rescheduling is self-explanatory. Resource alternative is making resource more available, while incurring more costs. (Zhu et. al., 368) In the case of this project, team members will put in extra work on Saturdays or other down times in order to recover from disruptions. The last resort is activity cancellation, which would result in dropping of features in order to reduce workloads and meet deadlines. Activity cancellation will be avoided at all costs.

**6.1 PERT graph**

Use Visio to draw a PERT graph for major tasks, showing task names, task dependencies, and elapsed time allocated to each task. This must be pasted into the MS Word document, so do not make it so detailed as to be unreadable.

**6.2 GANTT chart**



**6.3 Project Deadline**

|  |  |
| --- | --- |
| Task Name | Completion Date |
| Project description and elevator talk | Mon 2/16/15 |
| System specification / Project plan and classroom presentation | Mon 2/23/15 |
| Logic flow diagram | Tue 2/24/15 |
| Test-case list | Tue 2/24/15 |
| Increment 1 plan | Fri 2/27/15 |
| Increment 2 plan | Fri 3/27/15 |
| Midterm report | Wed 4/1/15 |
| Increment 3 plan | Fri 4/17/15 |
| Final report | Mon 5/4/15 |
| Create test data and environment | Sat 3/7/15 |
| Build logic flow diagram of data within program | Sat 3/7/15 |
| Construct Invoice object | Wed 3/11/15 |
| Construct Settings object | Wed 3/11/15 |
| Construct Finder object | Mon 3/16/15 |
| Construct Results object | Wed 3/18/15 |
| Construct Exporter object | Sat 3/21/15 |
| Construct interface allowing input of search criteria | Mon 3/30/15 |
| Construct interface for adjusting search and export settings | Mon 3/30/15 |
| Construct interface to view and operate on results | Tue 4/7/15 |
| Construct interface for filtering and sorting results | Mon 4/13/15 |
| Link the graphical interface with the program | Wed 4/15/15 |
| Create more use and test cases | Sun 4/19/15 |
| Create a test database | Mon 4/20/15 |
| Create queries to retrieve full filenames | Tue 4/21/15 |
| Modify search functions and objects to accept query results | Sun 4/26/15 |
| Present final product to PPG / Deploy product / Train users | Mon 4/27/15 |

**7. Risk Analysis**

The impact rank is from 1 to 5 (5 being catastrophic impact, 1 being negligible impact). RMMM stands for risk mitigation, monitoring, and management. It explains efforts KRSS will take to mitigate the probability of the risk occurring.

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Probability** | **Impact** | **RMMM** |
| Time loss due to other classes | 0.85 | 2 | Flexibility with work distribution among team members and purposeful time management |
| Team member not contributing | 0.02 | 2 | Scheduled group meetings and work checkups |
| Gaps in critical knowledge of C# | 0.1 | 3 | Member to member honesty about C# hang-ups and frequent use of online tutorials |
| Software incompatibility issues with PPG’s system | 0.35 | 1 | Communication with PPG sponsor and early backend testing with the actual system |
| Losing a team member | 0.01 | 1 | Redistribute work and reevaluate goals with Sponsor if necessary |
| Difficulty meeting or communicating with Sponsor | 0.5 | 2 | Ensure members have open schedules for sponsor meetings and communicates |
| Change in delivery date | 0.1 | 2 | Plan sprints to complete project before the sponsor’s deadline |
| Poor documentation on the part of the team | 0.4 | 3 | Schedule mutual documentation evaluations throughout each sprint |
| Application will not meet expectations of the Sponsor | 0.2 | 1 | Acquire strong mutual understanding of final application prior to beginning construction |
| Sponsor augments final application expectations | 0.15 | 2 | Retain record of sponsors expectations for the final product |
| Project is too small to meet class standards | 0.3 | 1 | Correspond with Dr. Yeager before beginning construction to ensure the success of the project |

**8 Bibliographical references**

Insert here a list of references, including both software engineering references and application-area references. This should include your textbook, at least two additional non-web sources, and at least two web sources. Use the MLA style guide (<http://www.lib.washington.edu/help/guides/44mla.pdf>) to determine proper form, for all categories of sources including books, articles, and web sites.