SigmaTEK Integration and Data Sharing

By

Randall Schneider

Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfillment of the Requirements
for
the Degree of Bachelor of Science
in Information Engineering Technology

University of Cincinnati
College of Applied Science

March 2006

SigmaTEK Integration and Data Sharing

By

Randall Schneider

Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfillment of the Requirements
for
the Degree of Bachelor of Science
in Information Engineering Technology

Randall J. Schneider	Date
Prof. Robert Schlemmer, Faculty Advisor	Date
Patrick C. Kumpf, Ed.D. Interim Department Head	Date

Table of Contents

Section **Page**

List of Figures

List of Figures (Continued)

Abstract

SigmaTEK International desires a solution that would make all upfront processing of data more efficient and enable them to build upon the solution for future improvements. The answer was the development of **S I D S (S**igmaTEK **I**ntegration and **D**ata **S**haring). The developed applications and database conversion solutions will enable SigmaTEK to efficiently and easily share data among current independent business systems that span Accounting, On-Line Helpdesk and internally developed *Order Entry*. It will also bring to SigmaTEK a level of automation to insure data integrity and eliminate redundancy.

This application will be used internal to the business. All users are current and future employees. Two of the applications will run as scheduled tasks and written in C#.net . Users will be unaware of the applications existence since the data will be changed and monitored by the application.

One solution consists of changes to an existing HelpDesk ASP site to assist in scheduling and automating day to day operations currently done very manually. The application runs on an Access backend and will be promoted to run on SQL Server.

Two other applications will come in the form of windows forms and will be written in Visual Basic. The user will utilize this interface between the application to insure that the data for the most sensitive data is accurate and complete. It will integrate back office data applications for Accounting (*QuickBooks*) and *Order Entry (OE)*. *OE* is a internally designed Interbase application for recording purchase orders and upholding SigmaTEK software security.

This overall solution will eliminate duplication of work through automation through a loosely connected architecture, streamline workflow and improve internal communication through real time information transfer. It will also result in a level of data integrity that would have been nearly impossible otherwise.

SigmaTEK Integration and Data Sharing "S I D S"
Business Systems Integration and Workflow Automation Project

Randall Schneider

1. Statement of Problem

The purpose of this project is to integrate and combine the SigmaTEK Corporations existing business systems. These applications supply much needed data to operate the business on a daily basis. It supplies data for accounting, order entry and other important customer specific data involving a HelpDesk ASP site which is supported off of the same data.

Since the current applications are independent of one another it creates an abundance of redundancy and manual hand-offs. It does not allow for sharing of data in a timely manner which lengthens the time needed to relate customer quotes and deliver product and impedes ash flow as well. The systems were purchased independently by their respective departments as the best tools for their needs.

SigmaTEK has realized the inefficiencies of their current system and looked elsewhere for solutions with all possibilities pointing to one fact. After the purchase of a new integrated business system, customization would still be necessary to integrate to the use the current SigmaTEK software security for licensing which is now supported by their *Order Entry* System.

SigmaTEK is a small business that develops software. It should be noted that it takes little to keep a business like theirs running as long as they can utilize a strong CRM system to collect and develop prospective customers and be able to bill and collect in a expeditious manner. As long as they stay ahead of competition by supplying a superior product and can effectively sell that product they need little to

remain competitive. The company already runs fairly lean and with the help of some integration they can become more efficient and operate even leaner in their day-to-day data flow.

I have reviewed the companies' internal processes and the current business applications and this project is a very good solution for their needs today and the foreseeable future. There is little wrong with having independent systems to run each area of the business. In some cases it makes good business sense to let each part of the business utilize the software that they feel will make them most efficient at their individual jobs. It is easier than ever to loosely connect these soft wares and a growing number of companies are taking advantage of these loosely connected developments to create business applications. If you do not change the core objects of the software then upgrading and other enhancements are not an issue.

2. *Description of the Solution*

This solution will prepare SigmaTEK for the future and take care of current needs as well. It should integrate the latest technologies and be more accommodating to the solutions that they have currently in place.

The solution will decrease the data redundancy and create a higher level of data integrity.

The solution will be used internally to the business on the existing network platform. The first change will come to the companies Helpdesk ASP database. An application change will be needed for the ASP HelpDesk Site as a request by

SigmaTEK in order to bring the system to something that will solve scalability issues and add a more robust database backend to their architecture.

A solution will also be implemented to perform the most time consuming and non-value oriented work to an automation level. It will schedule resources and monitor activities pertaining to the creation of Post Processors SigmaNEST Software, Special training needs for customers and Customer Start-Ups for the HelpDesk. Customer Start-Ups are calls created to the HelpDesk to check on new customers and their purchases. It functions as a customer satisfaction monitor. It also helps SigmaTEK know when an Order has been successfully delivered and completed to the customer's satisfaction.

Another element to this solution would be a link between their accounting *QuickBooks* and the current *Order Entry* System. Currently customer information is brought into SigmaTEK through their Customer Relationship Management (CRM) Software to collect the customer data for the entry of purchase orders. In order to invoice the customer the same data must be entered a second time into the current accounting software, *QuickBooks*. The likelihood of errors and the redundant steps are causing an undue burden on the staff and slowing up their ability to speed up cash flow.

During the evaluation process of their current process, these bottlenecks were found to be the most beneficial to SigmaTEK to be earmarked for elimination. Any solution would need to solve these issues.

The development and implementation of S **I D S** (SigmaTEK **I**ntegration and **D**ata **S**haring) is the answer. The executable applications and developed database solutions

will enable SigmaTEK to easily share date and create a level of automation among the current independent business systems. It will increase productivity and push data integrity.

2.1 User Profile

These applications will have two major types of users.

- Administrators
- End Users

2.1.1 *Administrators*

Administrators will have knowledge of all of the current systems Front End and Back End Products. They will be expected to know or have at least some familiarity with the technologies listed below in order to fully support the system into the future.

- C#.Net
- Visual Basic
- Database Connectivity, ADO.Net
- Database Administration
- Delphi - Interbase Server
- SQL Server
- ASP, VB Scripting
- XHTML, XML

They will also need to familiarize themselves with all documentation supported in this subsequent paper. The contents herein include all data flow maps and all logical elements that went into the development of the solution and internal system schematics as well. The purpose of this familiarization is to trouble shoot situations that may arise in the future and to insure that elements are not changed in the business system without thoroughly thinking through what ramifications or impact those changes may have on these applications.

2.1.2 End Users

All users are current and future employees. The users will already be using some form of the applications in which the solutions will be operating such as *QuickBooks*, *Order Entry* and HelpDesk. They are typically the administrative personnel and the Accounting staff.

Users of *Order Entry* will be trained on the new procedures for entering purchase orders. The procedure has changed little but the user will need to be aware of their impact on the new solution.

Users of *QuickBooks* need not know anything more than already known about *QuickBooks* since the impact will be little to none on data entry except that the user will not longer need to enter invoices for items entered as purchase orders in *Order Entry*. The user will need to know what not to do in this circumstance.

There will be training needed for the user of the OEToQB Application. The user will be trained on the minimal interface between the user and the application.

Users of the HelpDesk will need to be instructed to the new functionality and the impact that it will have on their day-today activities. They will be instructed as to what work would be moved off of their current list of requirement and how the new solution will be automated, thus eliminating past activities.

2.2 Design Protocols

SigmaTEK currently has a Microsoft Access database supporting the back end of the ASP Application. This will be changed to SQL Server 2000 and will run as if no change has occurred to the user.

One solution will be enhancements to the HelpDesk ASP Site. They will be written in VB Scripting and incorporate ASP Technology. The changes will use color coding and incorporate visual methods for making work more organized and easily viewable. It will make users more focused on what is at hand and not work coming in the future. It will also mean that other site pages be created for scheduling of resources and elimination of old email methods for sending notifications of orders and scheduled work.

Two solutions will be developed are part of a Visual Basic Windows Form for the creation of customers in *QuickBooks* from the internally developed

Delphi-Interbase (*Order Entry*) solution for the creation of purchase orders. The solution will migrate customer information from *Order Entry* to *QuickBooks* and create invoices from purchase order data reducing the likelihood of errors and eliminate the redundant entry of invoicing.

Two applications will run on the server as scheduled tasks and the users will be unaware of the applications existence. These solutions will come in the way of business logic changes to an existing HelpDesk ASP site and will be written in C#. It will assist in scheduling and automating day to day operations currently done very inefficiently and manually. These functional changes will drive a level of organizational scheduling of vital operations within the company and assist all users with a more visual means of monitoring their day-to-day work. It will schedule the companies Helpdesk for customer start-ups and create scheduling items within the helpdesk for internal resource planning for the creation of customer requirements such as post-processors and training. It will implement a visual method of color coding items which are most vital by due date.

3. Deliverables

In order to provide SigmaTEK with the proposed solution it is considered necessary to deliver the following solutions that were brought to light in the design and fact finding stage of this development.

The OEToQB Import, HDScheduler and the OEWatch applications will be a three stage process. Orders will be placed in Order Entry. All new orders will be set to default in the POTABLE to = 1. The process of each application will depend on the value or status of the HDSTATUS field.

HDSTATUS field in *Order Entry* will be used in the following manner:

Value of (**1**) = Staged to be processed by OEToQB for customer data transfer. OEToQB changes the Status after successful transfer to = (2).

Value of (**2**) = Staged to be processed by OEToQB for Invoice creation. OEToQB changes the Status after successful transfer to = (3).

Value of (**3**) = Staged for processing by OEWatch to run through each line item in the purchase order and create Customer Start-Up issues, Post Processor issues and Special Training issues in the HelpDesk. It also writes transactions to the C:\Temp\OEWatchTranactionLog.txt. It will write

all Error encountered to the C:\Temp\OEWatchErrorsLog.txt. OEWatch changes the Status after successful transfer to = (0).

Value of (**0**) = Processed through all stages.

3.1. Convert Microsoft Access database to SQL Server for ASP HelpDesk.

SigmaTEK wishes to convert their existing Microsoft Access database backend to their HelpDesk. They are currently moving a number of their other application to SQL Server so it's their wish to covert at this time.

3.2. HelpDesk Enhancements

Produce a useable tool to make use of visual methods to task resources and to help the user schedule their day. It would need little to no human input except for the entry of a Due Date for the HDScheduler if the current Due Date in incorrect.

- Establish New Departmental Site Information Pages.
  - New Post Scheduling
  - Unassigned Posts
  - Training Schedule
  - Customer Start-Ups
- Add Due Date column to Problems Table in the Database
  - This will be used to trigger the color code system.
- Write VB Scripting Code for ASP to present the pages above

- Write business logic using the Due Date to color code the work so that is presented in a more manageable and logical form for each department.

3.3. ***OEToQB* Import –** Visual Basic Windows Form Application**.**

Create an application that will allow the user to double checking the O*rder Entry* customer data prior to being populated to the *QuickBooks*. This application would not be automated at this point since it is vital to the cash of the company. Accuracy and reliability are crucial. It will be easy to use and allow the user flexibility for changing and determining what data will be transferred.

The application will also make use of buttons on the form to depict the process the user wishes to perform. It will also make use of form Menus at the top of the form to perform the same tasks.

This applications main purpose is to connect to the *Order Entry* Database and pull across all new Customers to *QuickBooks* if the following rules have been met:

- HDSSTATUS Field in *Order Entry* POTABLE table must be equal to 1.
- Must have dollar amount associated with purchase order.
- Must not be a United Kingdom order.
- Has not been flagged in *Order Entry* for Invoice Hold.
- P.O Status is Not of type Evaluation

After the customer is in *QuickBooks* the user can then push the *open Order Entry* purchase orders to create invoices for those orders in *QuickBooks*.

- Map the data flow from *Order Entry* to *QuickBooks*.
- Check for data type issues and field length differences
- Write logic map
- Scrub Data in *Order Entry* to clean up all the integrity issues
- Normalize *Order Entry* to lock down user input.
- Develop API application to automatically populate *QuickBooks* with information to be shared from OE (*Order Entry*) via COM and XML.

3.4. OEWatch for HelpDesk

Will watch for new purchase orders coming from *Order Entry* and will create HelpDesk Issues for scheduling, resources management and monitoring of job status and progress. This application will be written in C# and run as an executable in the scheduled tasks on the server. Once a day the new orders will be processed and depending on the Software module numbers ordered an issue will be created as an entry in the Problems table of the HelpDesk Database.

Some of the Issues that will be created are for:

- Special Training
- Schedule Customer Post Requirements
- Create New Customer Start-Up Calls

This application will no longer require a person to look over the *Order Entry* Line Items and create the appropriate scheduling data and internal requirements to meet customer needs.

3.5. HDScheduler System

Will watch HelpDesk Start-Up calls to Schedule the correct time and date to call the customer for the purpose of making sure that all is running well with the customer. *HDScheduler* runs as a scheduled task on the server after the *OEWatch* System has been run. It does so in order to schedule the current work coming in as new orders.

Will check to make sure all calls against a given customer are in the closed status and then schedule a date for call back. It will use a predetermined call back of two weeks after the last item has been closed. If no other issues arise for that two week period the due date will rise to the top and the issue will turn Green. Green means that it has reached a safe time to call or that it is current work to be done.

Color Coding System:

Red: Late work or is past due.

Black: Future work or has issues someone else is currently working and does not need the users immediate attention.

Green: Current work or needs to be addressed.

Users will no longer need to look at every call each week to see if there are open issues against a customer before calling to check if all is going well. This application will allow for more accurate tracking and save an estimated 2 hrs a day.

4. Design and Development

This section discusses the resources, time and costs associated with this project.

4.1. TimeLine

Listed below are the accomplishments and the challenges making up the time in completing the project on time and within budget. The challenges and unforeseen problems caused many delays and much lost time. Unforeseen issues with the internal Order Entry system did not help matters. The application was robust but the data integrity side of this was much an issue. Normalization in most cases did not exist so normalization had to be implemented and data scrubbed in order to make the full loosely connected architecture work.

4.1.1. Senior Design I – Accomplishments

During Senior Design I the follow accomplishments were achieved.

- Researched SigmaTEKs current operating conditions.
- Process Mapped the entire data entry process flow from the time an order is taken till the time the order is considered complete.
- Considered the needs of SigmaTEK as a company and the users.
- Investigated possible technologies for the solution.
- Debated the positives and the negatives of many possible solutions.
- Mapped the entire SigmaTEK Network Topology and data housing.

- Evaluated the Process map for the low hanging fruit to see where the most impact could be made.
- Pinpointed the key areas of impact.
- Investigated solutions for each key area because each area had different requirements and the timelyness of the information as well as the accuracy of the data were weighed out.
- Upgraded the HelpDesk Database to SQL Server.
- Changed ASP Scripts to point to the new Database.
- Test Off-Line
- Push to HelpDesk Site
- Added departmental pages and wrote all the business logic for displaying ASP pages in color coded manner using the Due Date.
- Test Off-Line
- Push to HelpDesk Site
- Created the proposal for the solution and created the presentation.

4.1.2. Senior Design II – Accomplishments

During Senior Design II the follow accomplishments were achieved.

- Met with company representatives to establish the requirements and map the business logic for the *OEWatch* System and the *HDScheduler*.
- Began developing both OEWatch and *HDScheduler*.
- Completed *HDScheduler* and began live testing.
- Completed *OEWatch* and began live testing.

- Prepared the Design Freeze documentation and presentation.
- Began Writing *OEToQB* Application.
- Began investigation of problems with writing application for the *OEToQB* Application.
- Could not get *OEToQ*B functioning with a .net wrapper. There was little to no documentation available on how to get this COM API functioning in .net.
- Changed Application language for *OEToQB* from C# to Visual Basic.
- Began the process of normalizing the *Order Entry* database in order to limit possible entries into *Order Entry* purchase orders.
- Added fields in *Order Entry* POTABLE to store data for QuickBooks crossover part numbers.

4.1.3. Senior Design III – Accomplishments

During Senior Design III the follow accomplishments were achieved.

- Completed the OEToQB Windows Application.
- Began Testing on local disconnected data.
- Started live data testing.
- Began final documentation of the applications and their functions.
- Finished final documentation.
- Presented the final project.

4.2. Budget

This project required the purchase of the items listed below. These items listed are the bare essentials since most of the other time and development tools will be either supplied by items already in the companies or my position.

Software		
Items	**Cost**	
QuickBooks Premier 2005		
5 seat license	$	1,400.00
latest API functionality.		
QODBC Driver		
2 licenses	$	300.00
Purpose: To be able to communicate directly to QuickBooks Data		
Total	**$**	**1,700.00**

Figure 1. Budget Data

5. Proof of Design

This section will discuss the deliverables and describe in detail how each deliverable was met.

5.1. Convert Microsoft Access database to SQL Server for ASP HelpDesk

From within Microsoft Access Database navigate to the top menu. Make the following selections.

<Select > **Tools**
Database Utilities
Upsize Wizard

The user is presented with the following screen. **(See Figure 2).** The user simply follows the instructions and it will convert the Access application to a SQL Server Database.

The wizard will also give the user information on errors that may have occurred and data type issues that may need to be resolved.

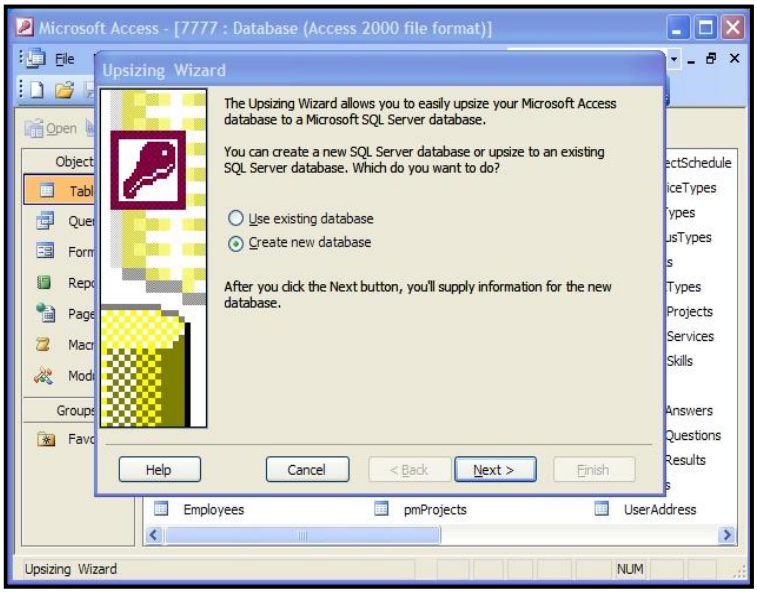

Figure 2. Microsoft Access (Upsize Wizard)

5.1.1 Convert Database to SQL Server Key Benefits

Future conversion of ASP site to ASP.net. It will also be a more stable database and allow more simultaneous users. SigmaTEK can begin taking advantage of the major benefits of SQL Server.

5.1.2 Convert Database to SQL Server Implementation Status

Deliverable Status: **(Complete & Tested & Implemented)** and has been in production for almost nine months.

5.2. HelpDesk Enhancements

This solution produced a useable tool to make use of visual methods to task resources and to help the user schedule their day. It would need little to no human input except for the entry of a Due Date. The Due Date will be used by the OEWatch System to set vital dates that the user needs to be made aware.

- Establish New Departmental Site Information Pages.
    - New Post Scheduling
    - Unassigned Posts
    - Training Schedule
    - Customer Start- Ups

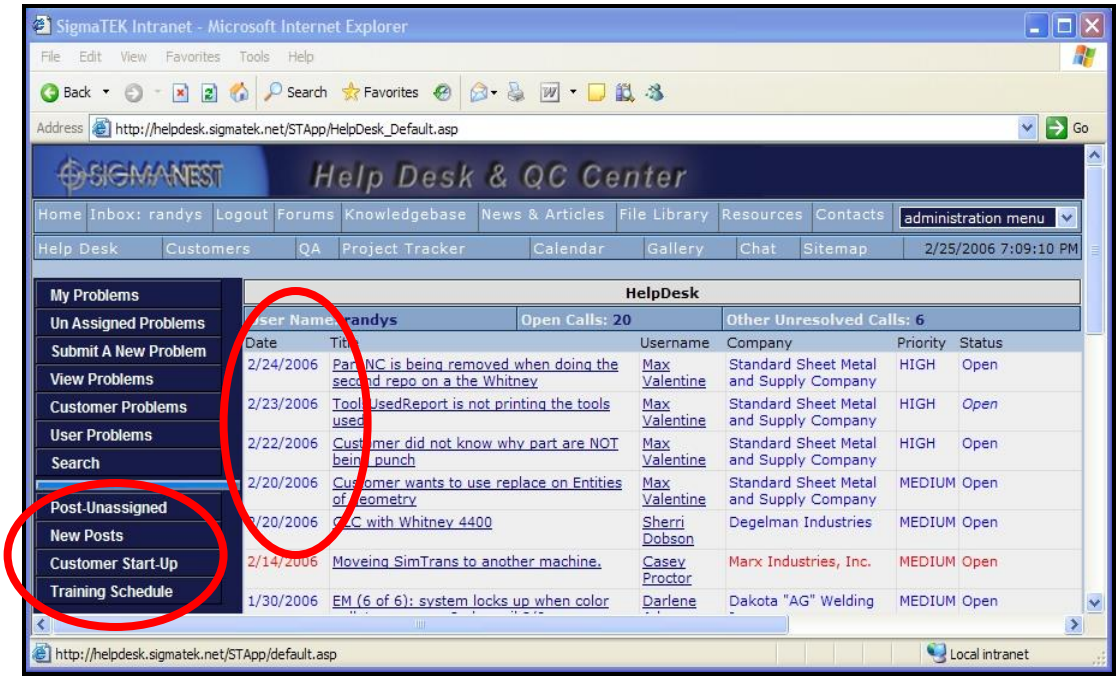

Figure 3. HelpDesk Users Main Page

- Add Due Date column to Problems Table in the Database
    - This will be used to trigger the color code system.
- Write VB Scripting Code to present the page above.

- Write business logic in VBScript for the ASP Site using the Due Date to **color code** functionality such that the data is presented in a more manageable and logical form for each department.

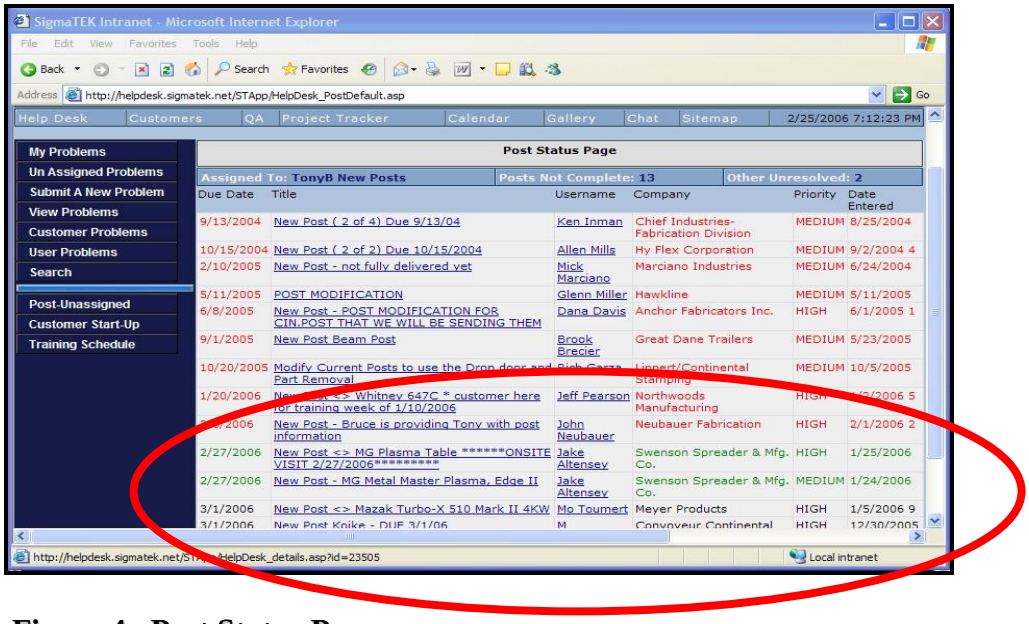

Figure 4. Post Status Page

Notice that the text for the issues is colored to make a more visual means of indicating its current priority.

Color Coding System:

Red: Text indicates Late work or is past due.

Black: Text indicates Future work or has issues someone else is currently working and does not warrant the user's immediate attention.

Green: Text indicates Current work or needs to be addressed.

5.2.1 HelpDesk Enhancements Key Benefits

The enhancements made to this ASP Site will bring to SigmaTEK a focus on the day at hand. The color coding system is a more efficient visual method of focusing on what is the current work or the work that the user should be focusing on today. The due date field can also be used for a notification or reminder of something the user wished to be on their plate for a future date or reminder to call a customer.

This color coded system is a hand down from this learners past experience in manufacturing. It is simple and if the disciplines are instilled in the users to complete the tasks daily the user need be focus on the few instead of the many.

5.2.2 *HelpDesk Enhancements Implementation Status*

Deliverable Status: **(Complete & Tested & Implemented)** and has been implemented into production for almost nine months.

5.3 OEToQB Import Windows Form Application

This solution was written in VB and is a Windows Form application. It will first push new Purchase Orders information coming from Order Entry (OE). Some of the information is dependant on whether this is s New customer or Current customer. It will also be transferring Purchase Order information such as Line Items from the same PO's to create Invoices via a COM API, using qbxml requests and responses to QuickBooks and the back to OEToQB.

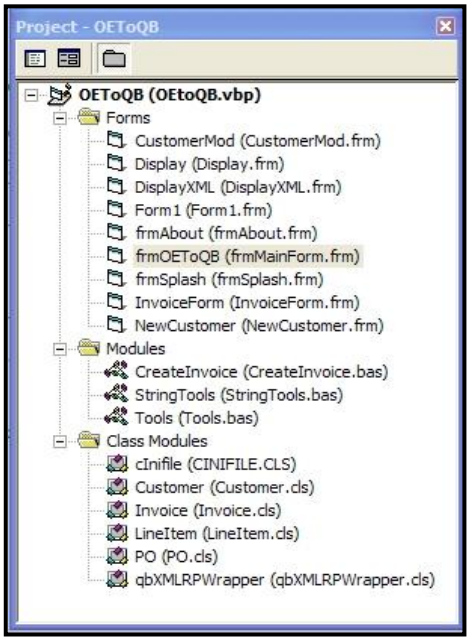

Figure 5. OEToQB VB Project

Figure 6. OEToQB Application Main Form

Below is a data flow diagram to show the use of HDSTATUS to determine the processing to be done on the data. **Figure 7** demonstrates a visual

explanation of the use of HDSTATUS in these applications and among other down-stream systems.

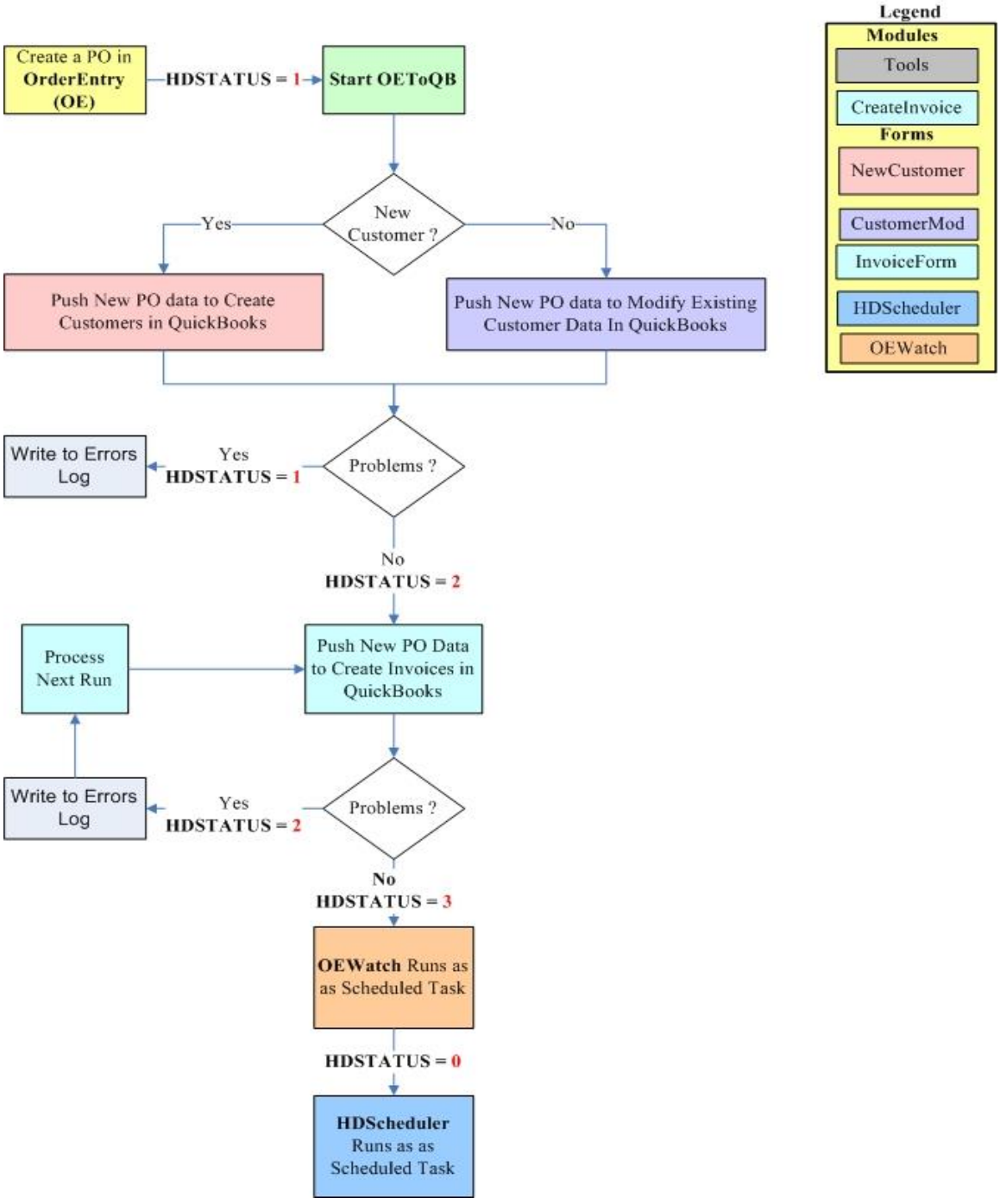

Figure 7. HDSTATUS Logic Flow

Note that all logic flow diagrams for this application and all other applications are located in the Appendices of this manual

5.3.1. Menu Selections

The user will be presented with a side bar menu to do one of two operations. They can either select to Push Customers or Create Invoice

5.3.1.1. Push Customers Button

When the user selects the Push Customers button the application will begin creating Objects and Collections of Object to be used in populating the needed data for the creation of New Customers in QuickBooks. The application will create a collection of PO's that store the purchase order data coming out of Order Entry (OE).

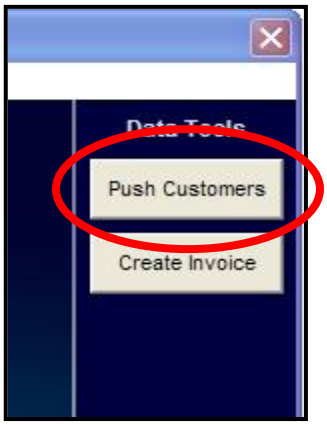

Fig 8. OEToQB Menu

The application will iterate through the collection and will create a selected PO object that identifies the current purchase order that the application is evaluating. This object is then passed to the New Customer Form for presentation to the user.

If the QuickBooks ID does not exist in the POTABLE under the QBID column in OE the application assumes that this is a new customer and will be

processed as a new customer xml request document to QuickBooks. It then uses the parsed response to pass back the associated QuickBooks ID to be stored for cross reference in Order Entry. The following screen will be presented to the user for new customers.

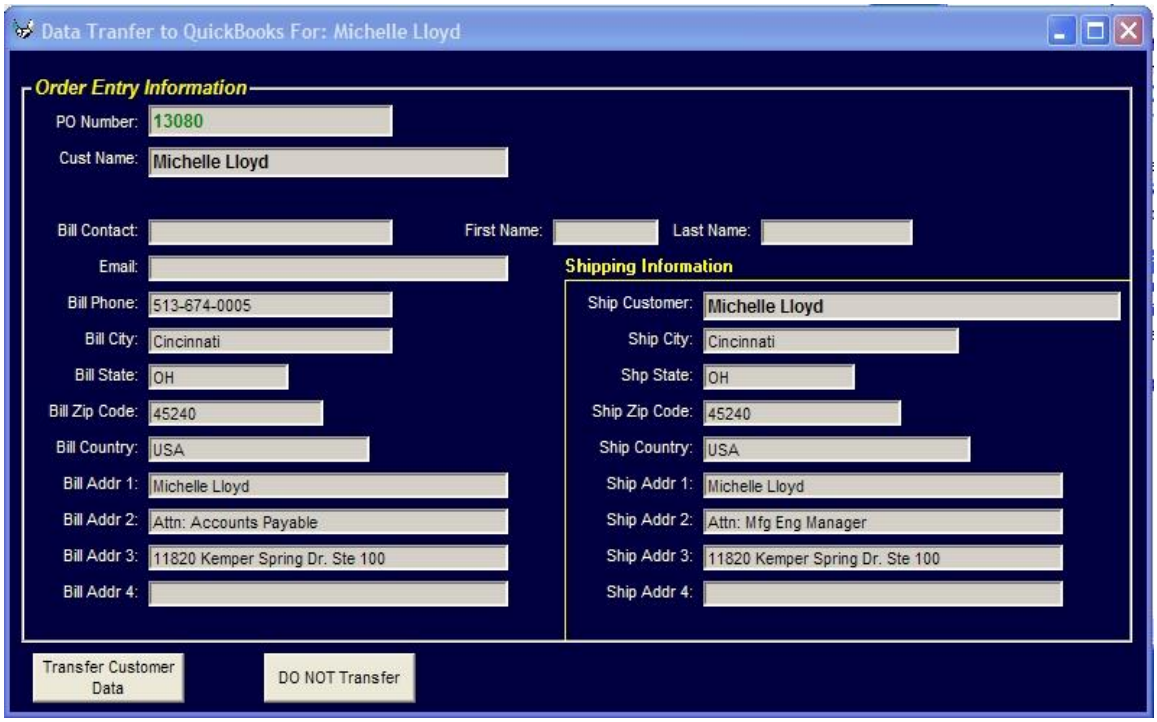

Figure 9. New Customer Form (OEToQB)

The user will be able to then select the 

button to push the current screen data and set the HDSTATUS = 2 in the Order Entry POTABLE. Changing this value to TWO will make the data accessible to the next method of *Create Invoices*. In order to create an invoice a CustomerID must first reside in the CUSTOMER table in OE in a column called QBID.

Upon successful completion of the transaction the user will be presented with the following message box.

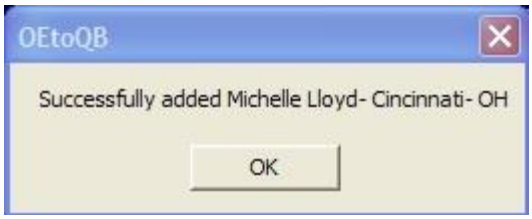

Figure 10. New Customer Added Message Box

The user can also select DO NOT Transfer and the application will keep the HDSTATUS = 1 and thus allowing the user to make modifications to the data in OE and then transfer the data at another time.

If the QBID is present to the application it is considered a current customer. This data will be handled in the application through a CustomerMod Form**.** The purpose of having an additional form for current customer is to allow the user some control over the data that will be populating the current customer data in QuickBooks. In **Figure 11** below, the users is presented data in two Panes. To the left, in **Yellow** is **Order Entry Info** and to the right, in **Green** is **QuickBooks Info**. Before presenting the data it is evaluated to see if the current QuickBooks data is different from that of Order Entry. If it is different, the data is presented to the screen in **Red** on the Order Entry Info Side. This alerts the user of the difference and allows the user to determine whether the change will be transferred to QuickBooks via the Update QuickBooks Button and the HDSTATUS in the POTABLE in OE will be changed to a Two and thus ready to process the Invoices.

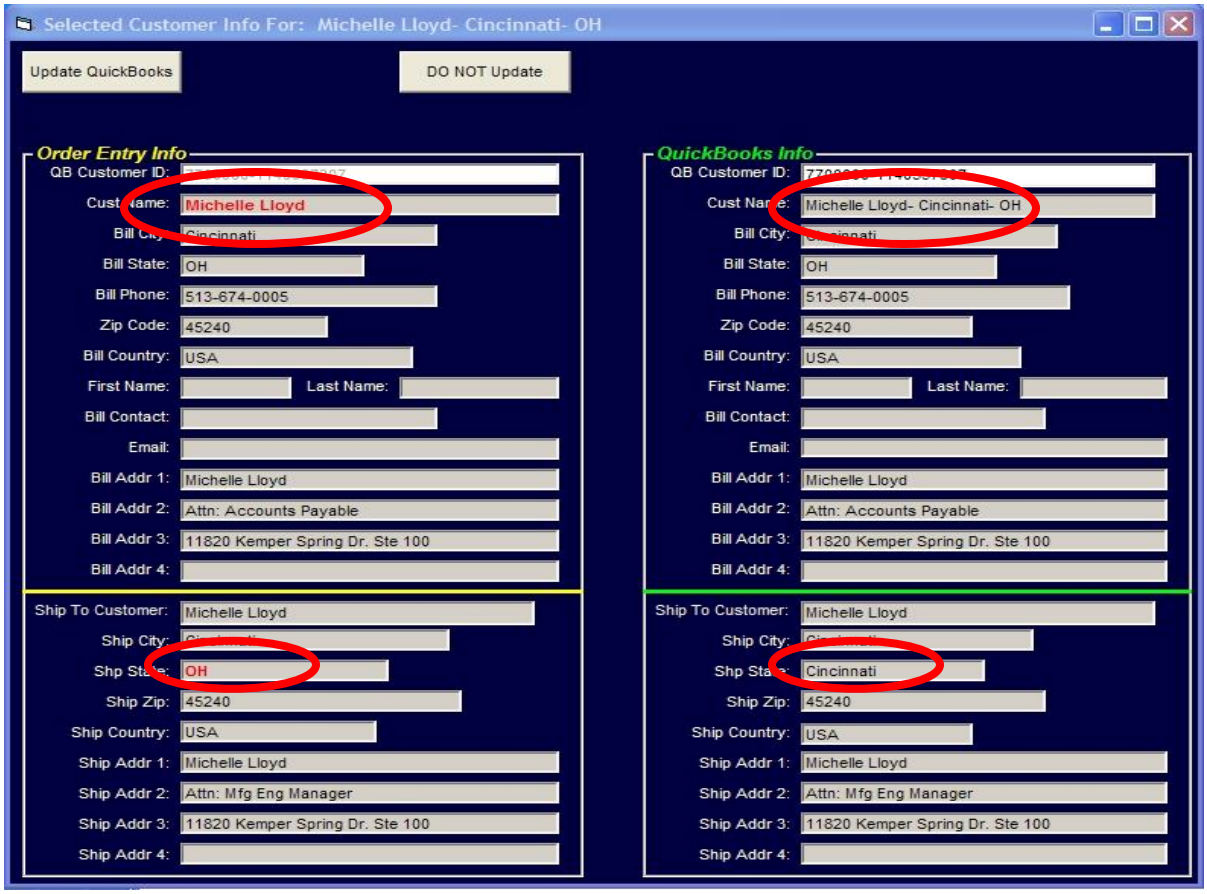

Figure 11. CustomerMod Form View

The user can elect to select the DO NOT Update button which will change the HDSTATUS = 2 and will not update the QuickBooks customer Data to match the data currently on the Order Entry Info side on the form.

5.3.1.2. Create Invoices Button

When the user selects the Create Invoice button the application will begin creating Objects and Collections of Object to be used in populating the needed data for the creation of invoices in QuickBooks. The application will create such objects as Line Items and in turn they will be stored in a collection

called Invoices and then stored in a collection object. It will do the same for PO's.

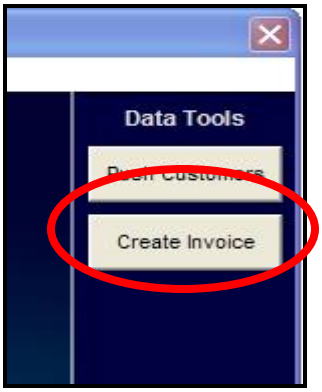

Figure 12. OEToQB Menu

The application will iterate through the collection and will create a selected PO object that identifies the current purchase order that the application is evaluating. It then finds the associated Line Items in the Items collection and creates a selected Items collection. Both of these objects will be passed to the Invoice Form for presentation to the user. **See Figure 13**. Right, for a view of the Invoice Form.

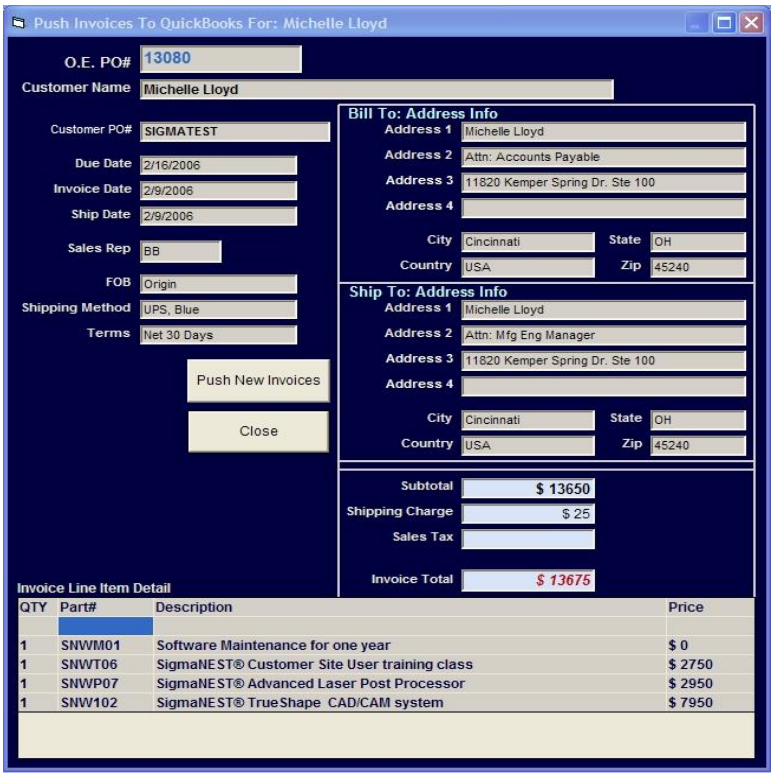

Figure 13. Add Invoice Form

5.3.2. OEToQB Error Handling and Transaction Logs

Errors during the use of this application will be thrown to the user and at least for the first year they will be written to the OEToQBErrorLog.txt file. The application will also record the transactions that take place during the use of this application file located in the C:\Temp\ directory on the server and are in a shared folder. The transactions will be written to the OEToQBTransactionLog.txt file. Both can be seen in **Figure 14** below.

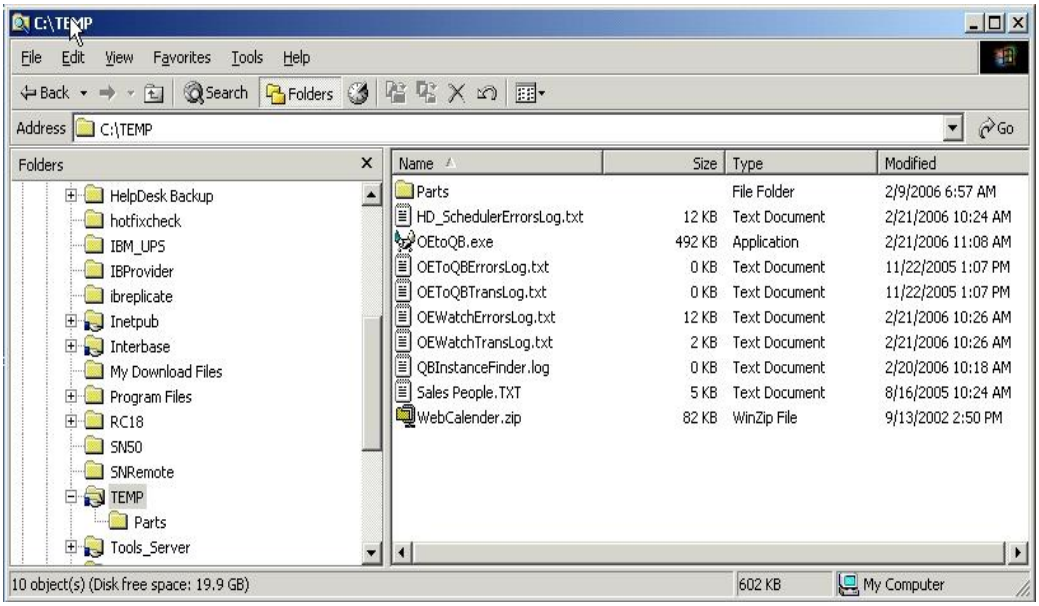

Figure 14. Errors and Transaction Log Locations

5.3.3. OEToQB Benefits

The benefits of this application numerous. Listed below are a few key benefits of having a loosely connected link between the independent systems.

- No redundant data entry
- Very high level of data integrity.
- Updated in one system and automatically transacted to the other.

- Prices need only be changed in one system

5.3.4. OEToQB Implementation Status

Deliverable Status: (Complete & Tested & Implemented) and has been implemented into production for almost nine months.

5.4. OEWatch System

The OEWatch system is a C# application that runs as a scheduled task on the server. It will process all new OE purchase orders with HDSTATUS = 3 from the Order Entry POTABLE. This status indicates that the data has been processed by the OEToQB application and is now ready for further processing and scheduling of resources and special needs. It will schedule such resources as Creation of Posts, Special Training Needs and New Customer Start-Ups.

The application will monitor the line items within the purchase order and based on those items a resource entry is added to the HelpDesk Scheduling ASP pages to notify the recourses of the new requirements and their delivery date (Due Date).

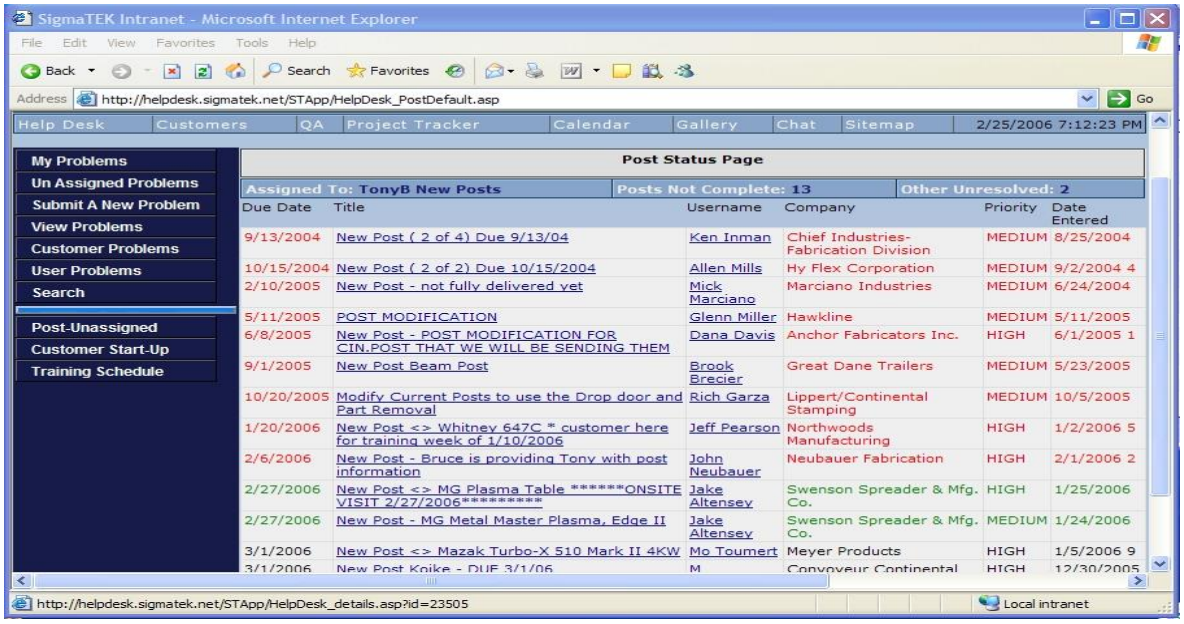

Figure 15. Post Status Page

Figure 15 above demonstrates the items as they would appear in the HelpDesk. Note that three other selections will display the data the in the same fashion. The other selections are shown in **Figure 16**.

Figure 16. HelpDesk Menu

5.4.1. OEWatch Error Handling and Transaction Logs

Errors during the use of this application will be written to the OEWatchErrorsLog.txt file. The application will also record the transactions that take place during the running of this application file located in the C:\Temp\ directory on the server and are in a shared folder. The transactions will be written to the OEWatchTransLog.txt file. Both can be seen in **Figure 17** below

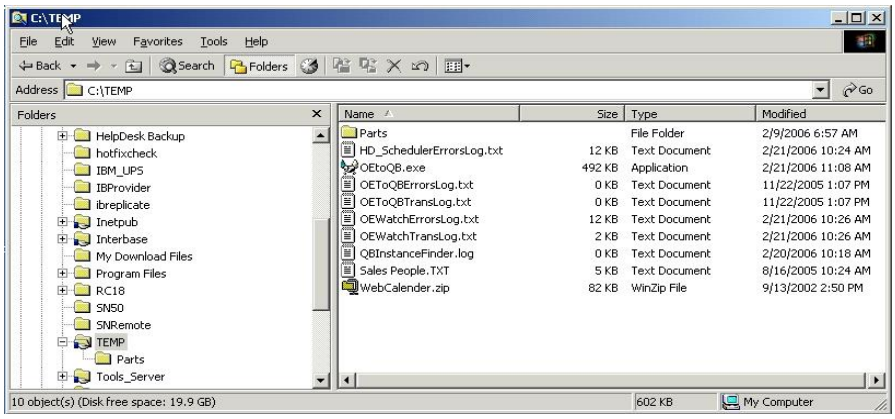

Figure 17. Errors and Transaction Log Locations

Below is an example of the OEWatchTransLog.txt file. The transactions are added each time the application runs and posts new items to the HelpDesk. It will post the date and time OEWatch ran and the Purchase Order Number and the type of Issue created.

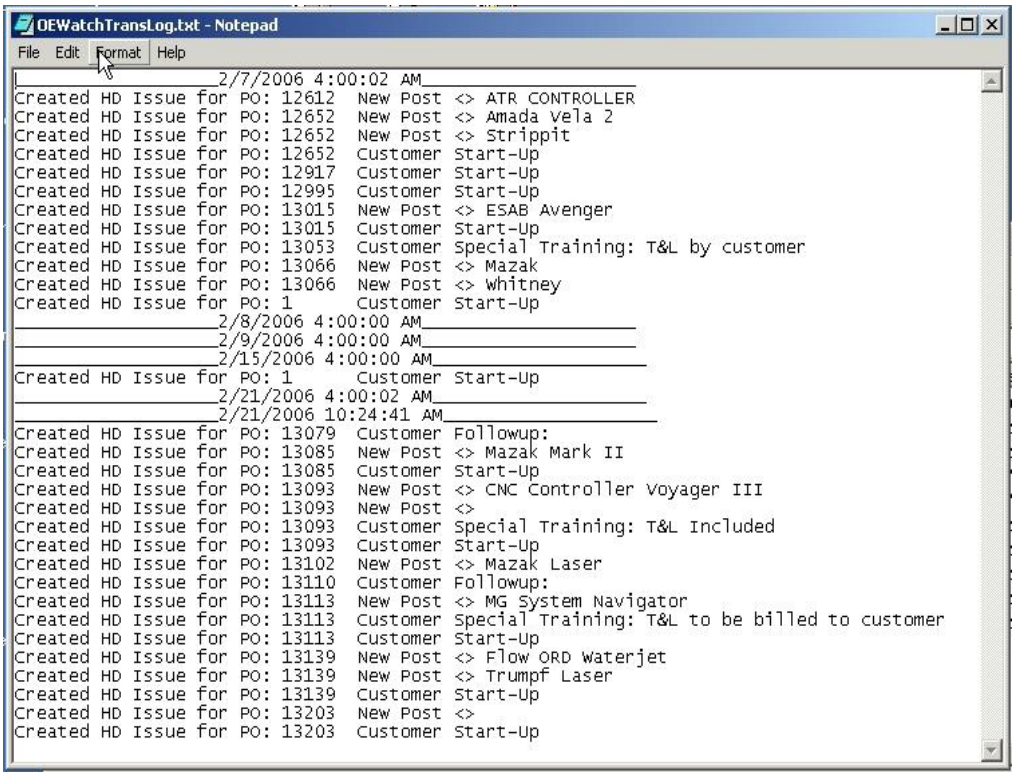

Figure 18. OEWatch Transaction Log

It is no longer necessary for the endless email notifications and forgotten resource assignments. The system watches the orders and pushes the data automatically each day.

5.4.2. OEWatch Key Benefits.

It is no longer necessary for endless querying of the HelpDesk ASP Site to check each day if all issues are closed against that customer. Since the query that the administrative personnel is executing is only good for that day, it is no longer valid the next day since a call could have been logged against the Customer Start-Up a moment after it was reviewed.

5.4.3. OEWatch Implementation Status

Deliverable Status: **(Complete & Tested & Implemented)** and has been implemented into production for almost nine months.

5.5. HDScheduler System

The HDScheduler system is a C# application that runs as a scheduled task on the server. Its function is to monitor Customer Start-Ups to notify the HelpDesk Customer Start-Ups Administrator when a call should be placed to the customer to see whether they are satisfied with the product and the initial installation and subsequent issues that may have risen during the start-up stage.

When a customer places an order through Order Entry a Customer Start-Up is created by the OEWatch system and is given a Due date for two weeks from the time of the order. If an issue or problem should arise and a call is placed to the HelpDesk for that customer the HDScheduler will continue to push the Due Date for the Customer Start- Up call until all issues related to that customer are closed. Once all issues are closed HDScheduler simply stops rescheduling the Customer Start-Up and the issue tied to the Start-Up simply makes its way up the priority list in the HelpDesk **Customer Start-Up** ASP page. Once the related issue turns Green the call should be made to the customer to get their sign-off that all is running well.

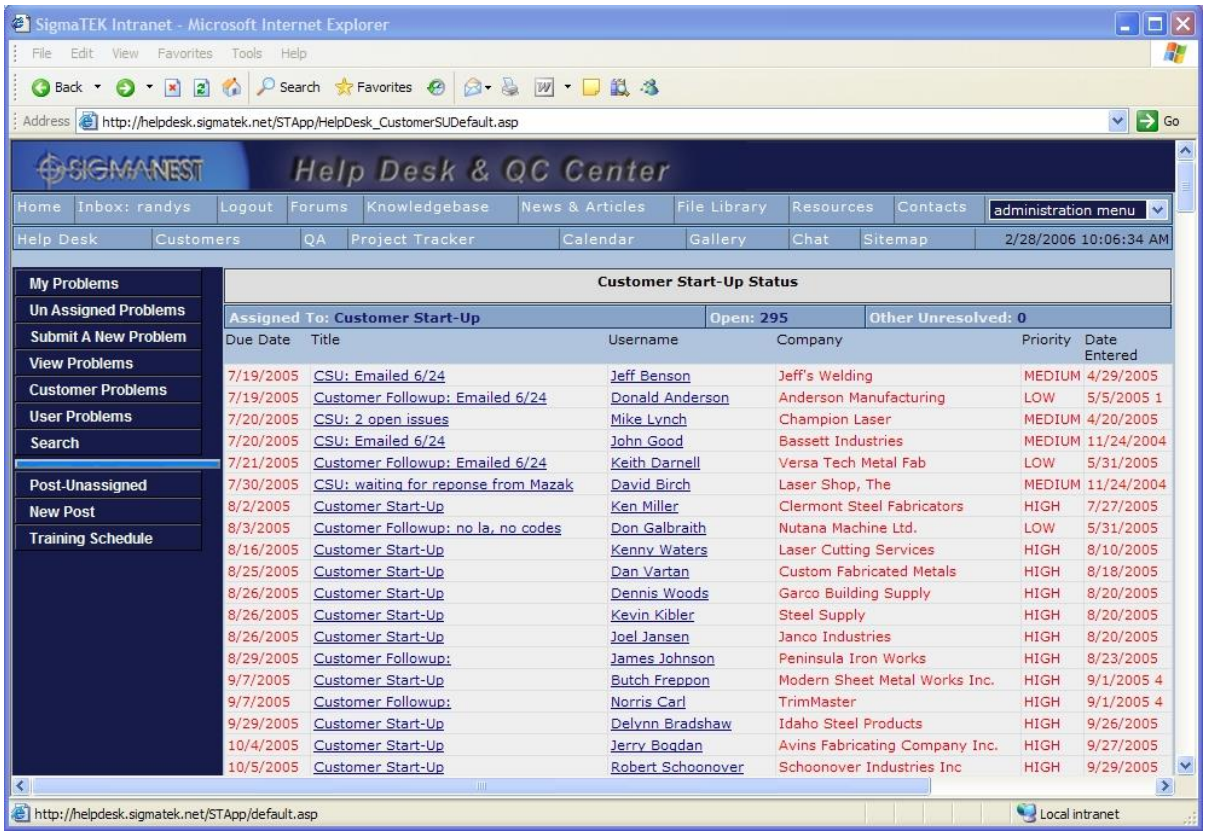

Figure 19. Customer Transaction Log

5.5.1. HDScheduler Error Handling

Errors during the use of this application will be written to the HD_SchedulerErrorsLog.txt file located in the C:\Temp\ directory on the server and are in a shared folder. The application will not record the transactions that take place during the running of this application due to the sheer number of transaction that will take place each day.

5.5.2. HDScheduler Key Benefits

It is no longer necessary for endless querying of the HelpDesk ASP Site to check each day if all issues are closed against that customer. Since the query

that the administrative personnel is executing is only good for that day, it is no longer valid the next day since a call could have been logged against the Customer Start-Up the same day as the last time it was reviewed.

Calls will not be made to a customer to see if all is well only to find out that there are current issues and the call turns quickly into a call to find out the status of the issues that are logged against that customer.

5.5.3 HDScheduler Implementation Status

Deliverable Status: **(Complete & Tested & Implemented)** and has been implemented into production for almost nine months.

6. Testing Procedures

In order to test this application off-line it was necessary to replicate the entire SigmaTEK data infrastructure. The different application could be tested completely off-line to the production application until all parties felt it was safe to test on live data. A simulation of the same live data was loaded to the application and then run local to the test machine for any possible issues.

Each application has its own transaction logs or error logs to make the administrator aware of any issues that may be causing problems on the system. At some point it would be a good idea to add the functionality for the application to email the administrator of any issues should arise.

To replicate the Order Entry information it was necessary to set up a duplicate Interbase server on the test machine as well as all data bases, stored procedures and customer scripts necessary to the replication of all Interbase and OE transactions. OE is also where the HelpDesk application which is a SQL Server ASP Site retries all of its customer information order data. The SQL Server application only acts as a back end to log issues therefore it was essential to install and setup SQL Server 2000 on the local test machine.

A fundamental element to the testing process was setting up the ASP Site on the test machine. It was necessary to configure IIS and copying all essential ASP pages to the IIS service. Since this application would be set up to mimic that of the existing system it was not key to the application to change much to the connectivity elements of the ASP Site. The test machine could be setup accordingly to allow easy implementation to and from the live applications.

The testing continued during the implementation stage. All issues that would arise would be emailed or logged and worked on immediately. Issues are still rearing their ugly head today but are on a very small scale. Most of these issues pertain to the OE and QuickBooks Application *OEToQB*. The majority of the occurrences are that of the integrity of the data coming from upstream OE. Before writing the application a great deal of data scrubbing and normalization had to occur before even thinking about a line of code between a loosely

connected architecture. We are on the final stages of locking down and securing the integrity needed for this to perform flawlessly. A great deal of time was spent just trying to trouble shoot the loose ends. As of March 3, 2006 all has been running without so much as a hiccup.

7. *Conclusions and Recommendations*

7.1. *Conclusions*

Before starting this project a meeting took place to discuss what SigmaTEK was looking for in this application. The scope and time requirements to fulfill the entire wish list of requirements was far too large for the time constraints on a college Senior Design project while working and traveling for the same company. The wish list was scaled back and the low hanging fruit was tackled first. The wish list was to create an application that would do all that a high-end product such as Great Planes yet work seamlessly with their current business system. This was not possible for the time given and the problems with the current data that we had to start with.

What they got however was a smaller answer to a big data problem and an enormous time saving mechanism to the people who use the system day in and day out. After process mapping the entire data flow from beginning of contact with a customer to the delivery of the product and the subsequent problem solving thereafter the applications within this project solve all of the data integrity issues. They also address data flow and wasted time as well as setting the basis for future

development of this application and customization without having to fix all the past data sins.

This application gives SigmaTEK data integrity between their Order Entry system for taking orders and their QuickBooks application for billing and accounting needs. It is all done somewhat seamlessly with little or no user inputs but to push a button. The application should save on an estimated basis of 1 ½ to 2 Hours a day for the data entry personnel and will cut down the mistakes made during redundant data entry from one system to the other. The time saved on just OEToQB application alone is almost ½ a man year or 500 hour annually.

Order Entry Watch (OEWatch) and the HelpDesk Scheduler will save another estimated 500 man hours a year and make better use of time and resources for more value oriented work.

Data within SigmaTEK is now flowing more efficiently. Data is now moving from application to application within hour not days and scheduling is taking place more efficiently than ever.

This application solution was far bigger that a student in a Senior Design could handle as a project without also working at that location. The stumbling block came one after another and there were far more problems with the current application than originally anticipated. This exercise has opened my eyes to the

process of research, design, testing and implementation than ever anticipated. There was no way to have understood all the pitfalls that this learner ran into during this experience and it therefore gave this learner a better understanding of how to better estimate future projects.

7.2. Recommendations

It is my belief that this application could be taken another step further and made a bit more automated when it come to OEToQB. This application is in its infancy and needs to be run in its current fashion until all has been functioning well for at least a month or two. The next step with this application would run *OEToQB* as a scheduled task and run off-line to normal day to day work much like the OEWatch system and the HD Scheduler. It can be automated to print the invoices automatically and thus there would be no need to see QuickBooks for at least one employee as an invoicing mechanism.

The foundation is now laid for the future for SigmaTEK. It is imperative that anyone taking over the maintenance of such a system understand the entire process thus to not break the current programming mechanisms or triggers.

It's also my recommendation that an automatic notification be emailed from the applications when something is wrong. This will be a somewhat easy

undertaking since the error handling systems are there but are just sent to a file versus sent to a mail recipient.

This learner would have used a different COM API on the OEToQB application for communicating to QuickBooks. The qbxml API seems to be underdeveloped and the older API is a bit more established and has fewer bugs. This learner found after developing the customer transfer portion of the OEToQB application that was originally written in C# with a .NET wrapper would not function and little to no information or literature on the issues behind its problems with the COM API. We then had to drop back and create the same application a second time in VB. It's this learner's opinion that going out and trying to get something working instead of going with what has already been functioning was a bad idea learned the hard way.

Appendix A

Project Time Line

Below is a timeline that was based on the development work needed to accomplish the implementation**?** This diagram also includes time during the spring quarter in which we were not in class but was still working on this project.

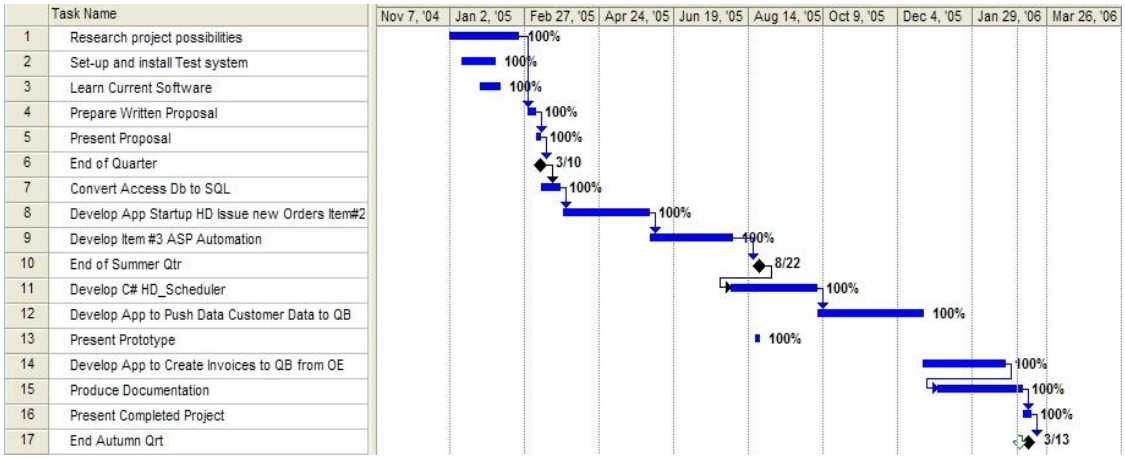

Figure 20. Project Time Line

Appendix B

SigmaTEK Topology Diagram

This diagram displays the current Topology. This was created in order to have a better understanding of where all the data was coming from among all the software used.

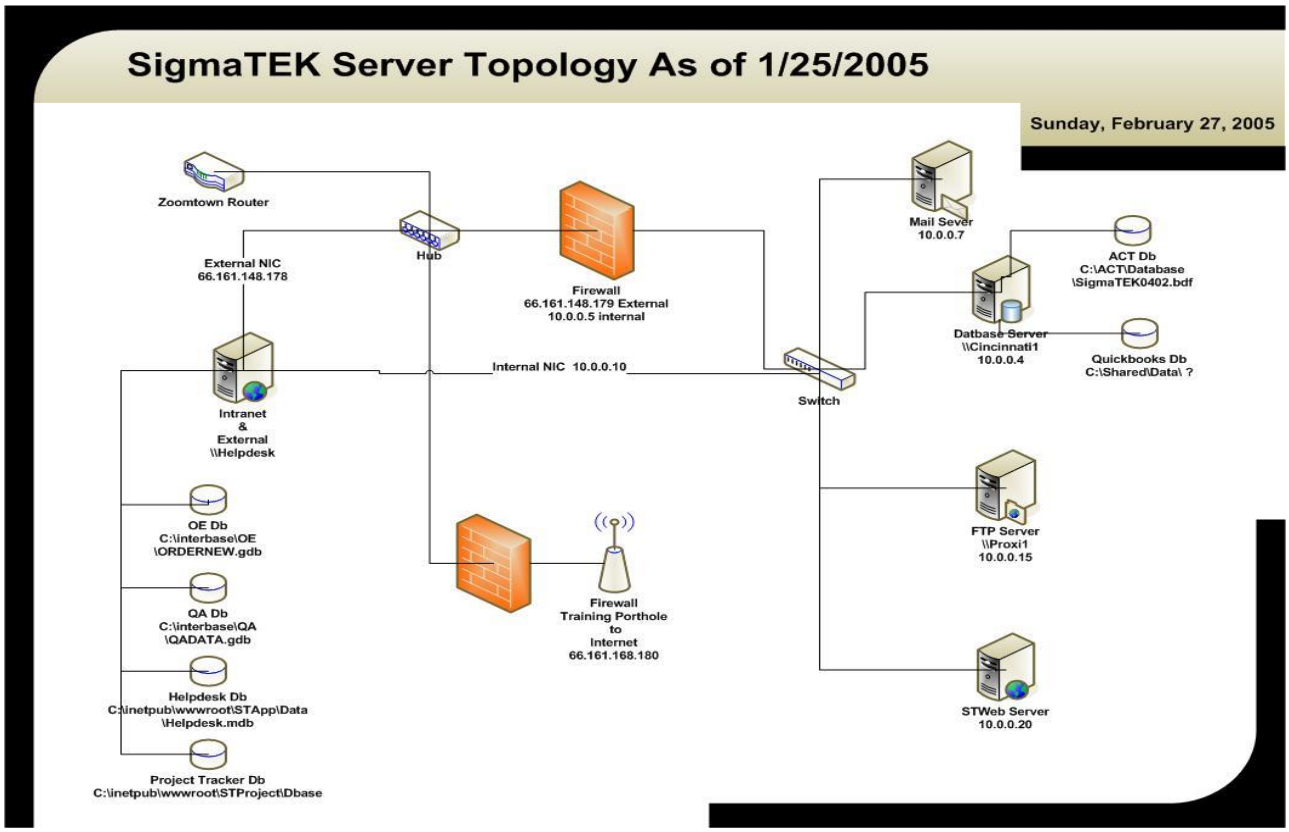

Figure 21. View of Topology Prior to Project

Appendix C

C1. Sales Order Process Flow – Original Process Before Implementation

This was the process flow diagram for the Order Entry and the flow of data to one person to the next. Notice the number of steps and the number of unique software's needed to conclude this daily process. We used this to pin point the areas which would be attached or marked for automation and would give the highest return of our time.

See Figure 23, next page for a process flow diagram displaying the process prior to the implementation of this project

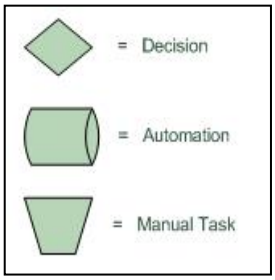

Figure 22 Task Legend

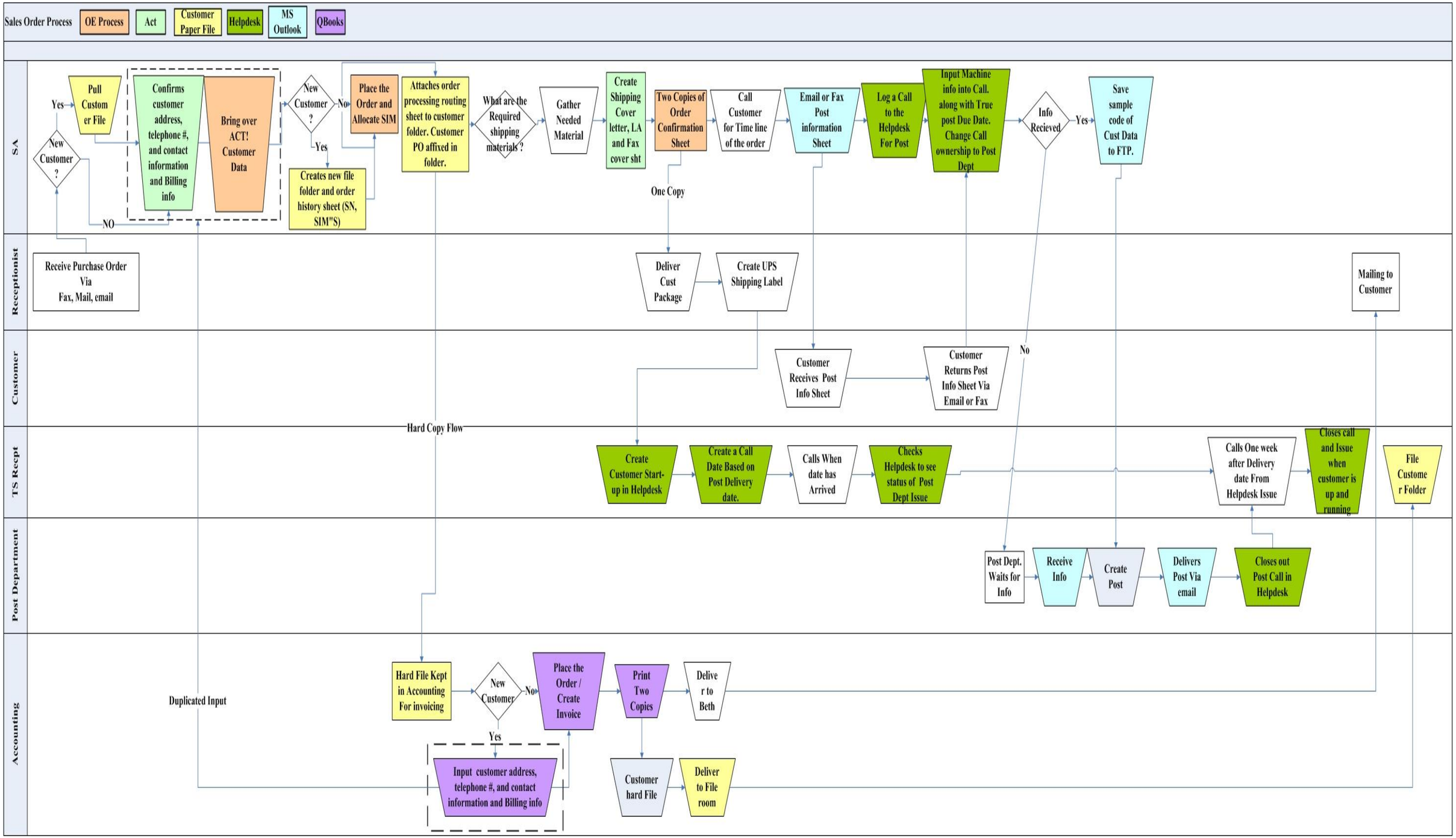

Figure 23. Sales Order Process Flow Prior to Project

C2 Sales Order Process Flow – Original Process After Implementation

This is the proposed process flow indicating the flow after implementation. The most notable elements of this are the automated processes indicated by the cylindrical objects. These would take away all redundancies and minimize the users need to interact with the business system on a daily basis. Some of the most menial tasks would be automated and a built in business logic would be programmed in to set forth a viable and repeatable business flow.

See next page for a diagram displaying this process. (Figure 24)

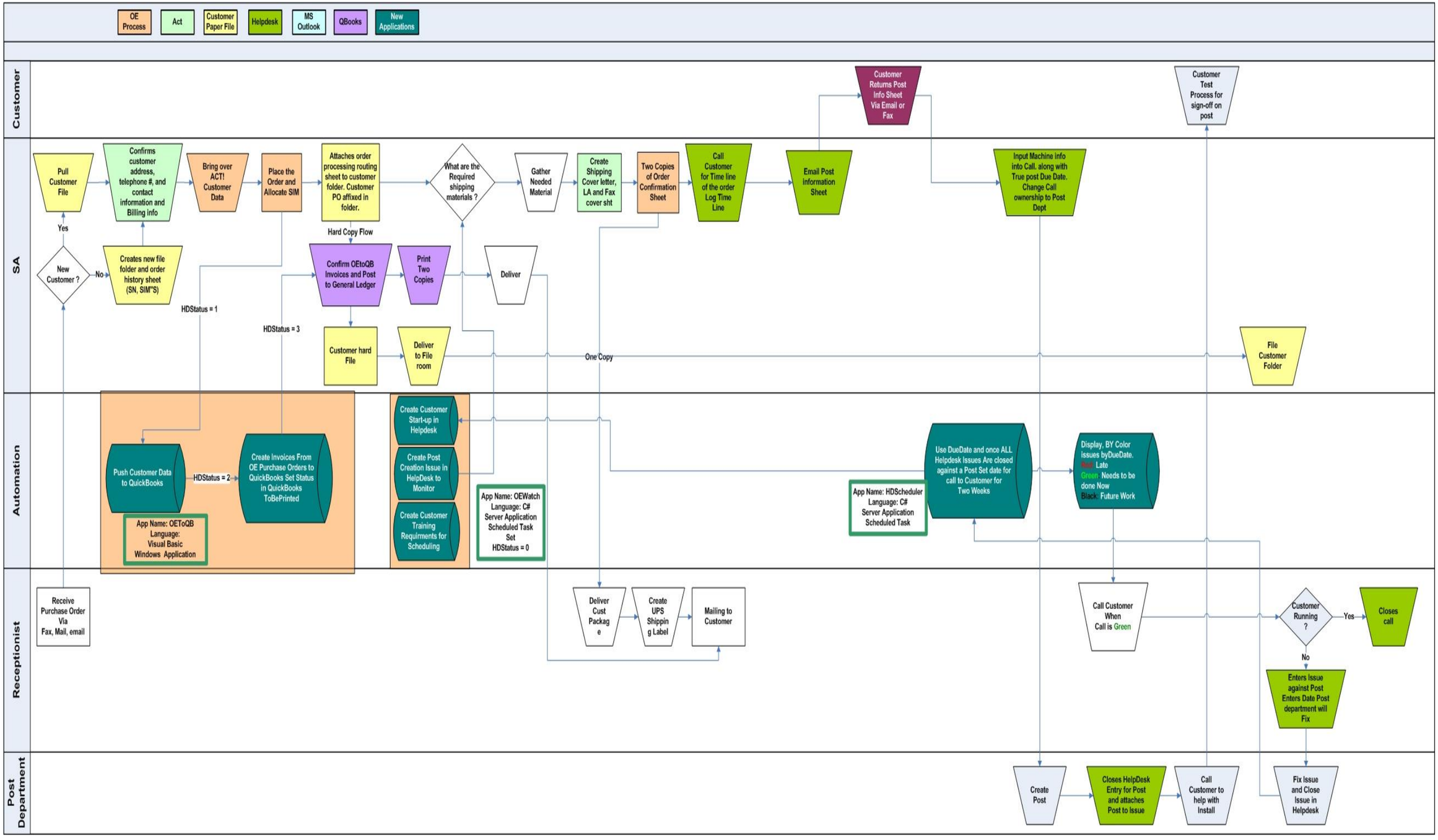

Figure 24 Final Sales Order Process Flow After Project Completion

Appendix D

D1. OEToQB Logic Flow for creating New and Modified Customers

The logic flow diagram below, illustrate the logic used when the user selects the Push Customers Button from within OEToQB.

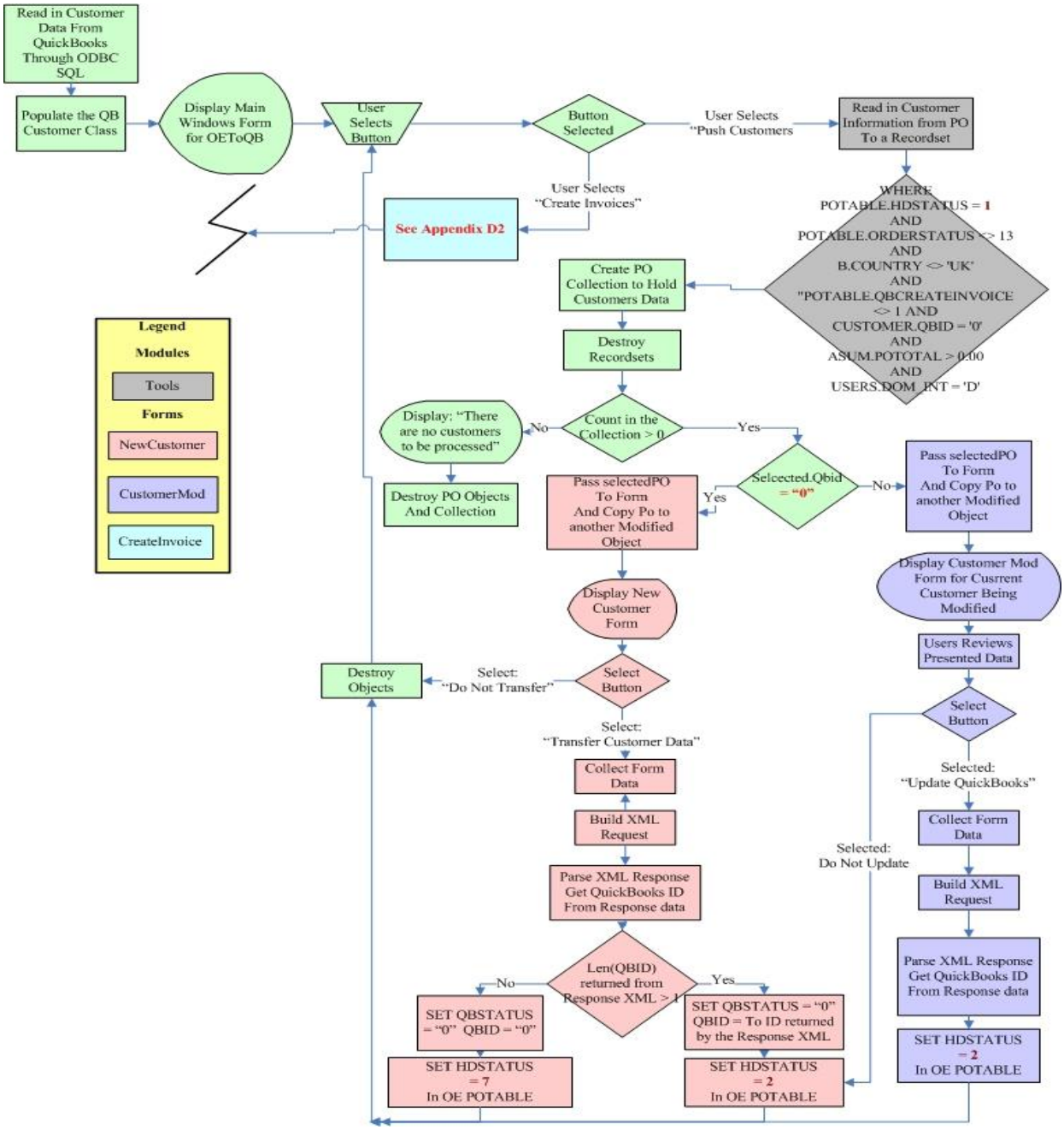

Figure 25. OEToQB - Logic Diagram for Creating New & Modified Customers

D2. OEToQB Logic Flow for Creating Invoices in QuickBooks

The logic flow diagram below, illustrate the logic used when the user selects the Create Invoices button from within OEToQB.

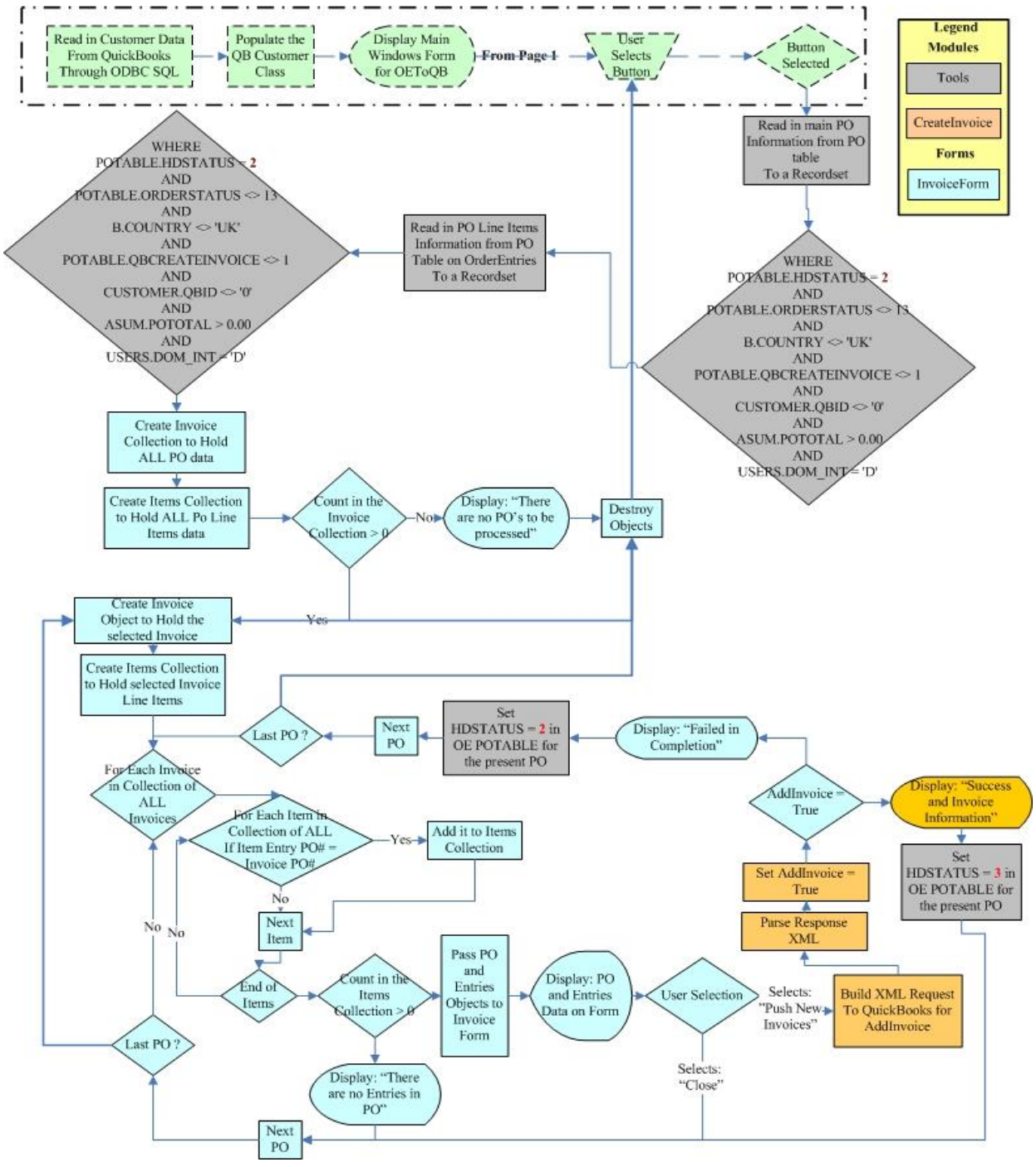

Figure 26. OEToQB - Logic Diagram for Creating Invoices

D3. HDScheduler – Logic Diagram

The logic flow diagram below, illustrate the logic used when the user selects the HDScheduler scheduled task application.

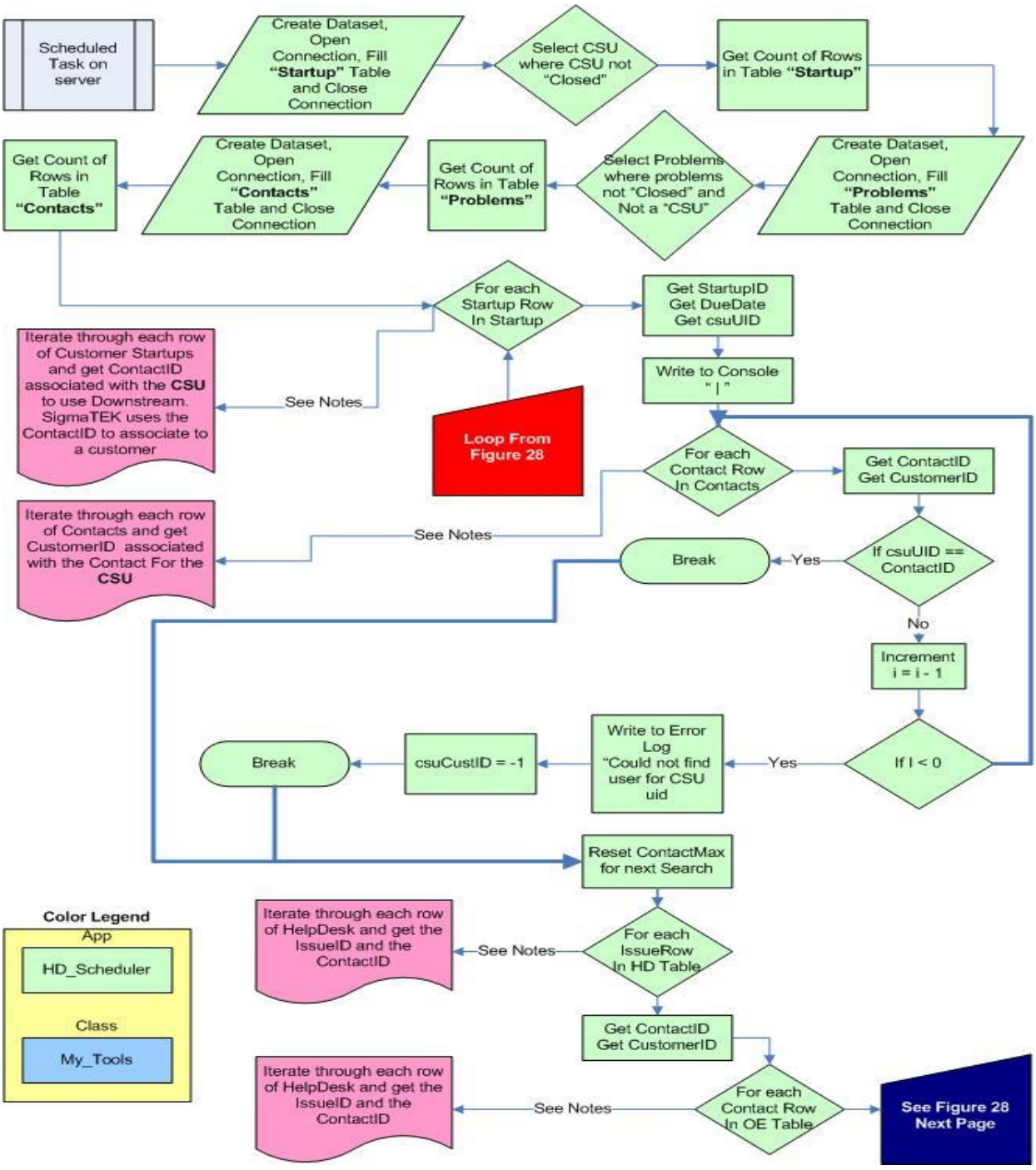

Figure 27 HDScheduler Logic Diagram 1 of 2

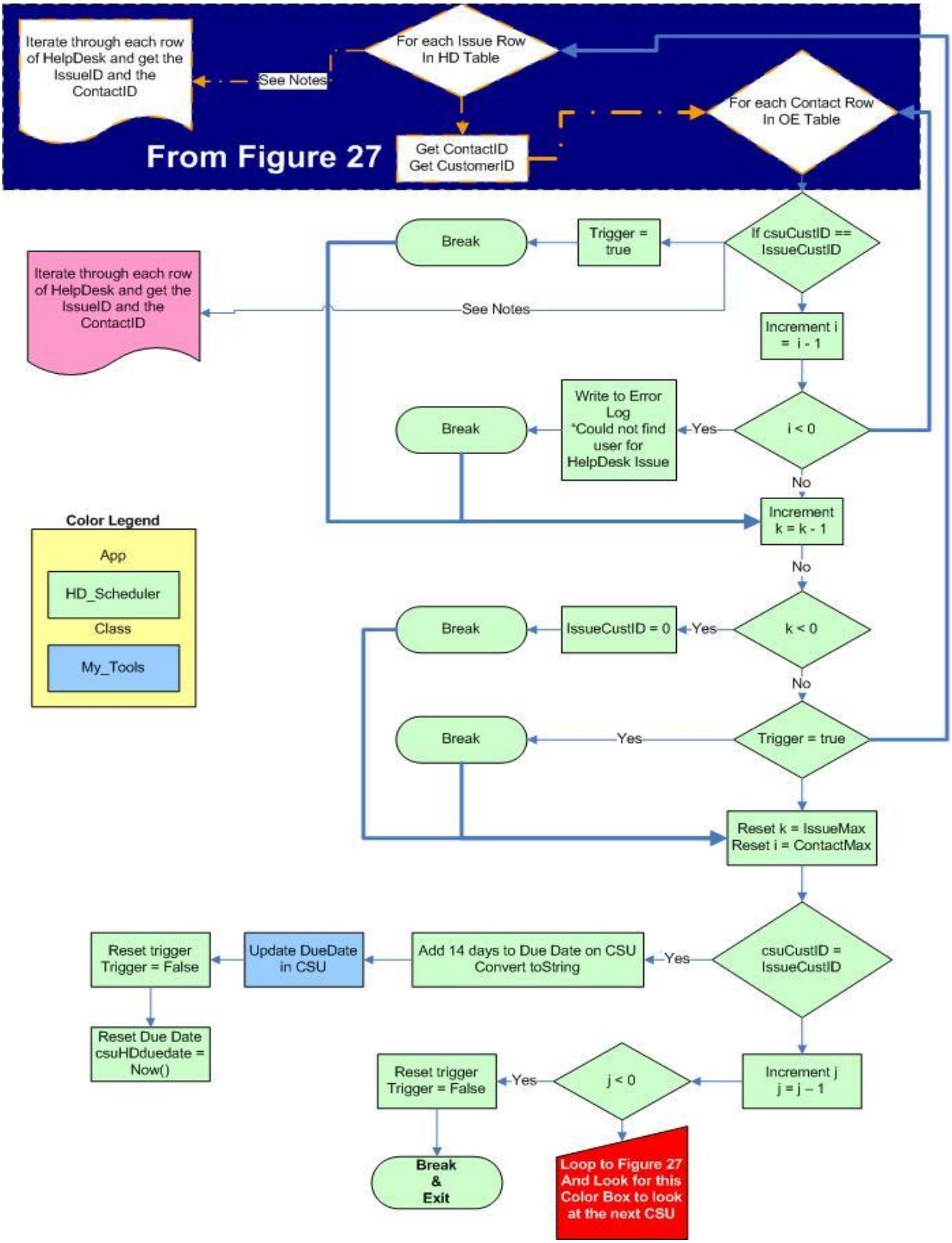

Figure 28 HDScheduler Logic Diagram 2 of 2

D3. OEWatch – Logic Flow Diagrams

The logic flow diagram below, illustrate the logic used when the user selects the OEWatch application.

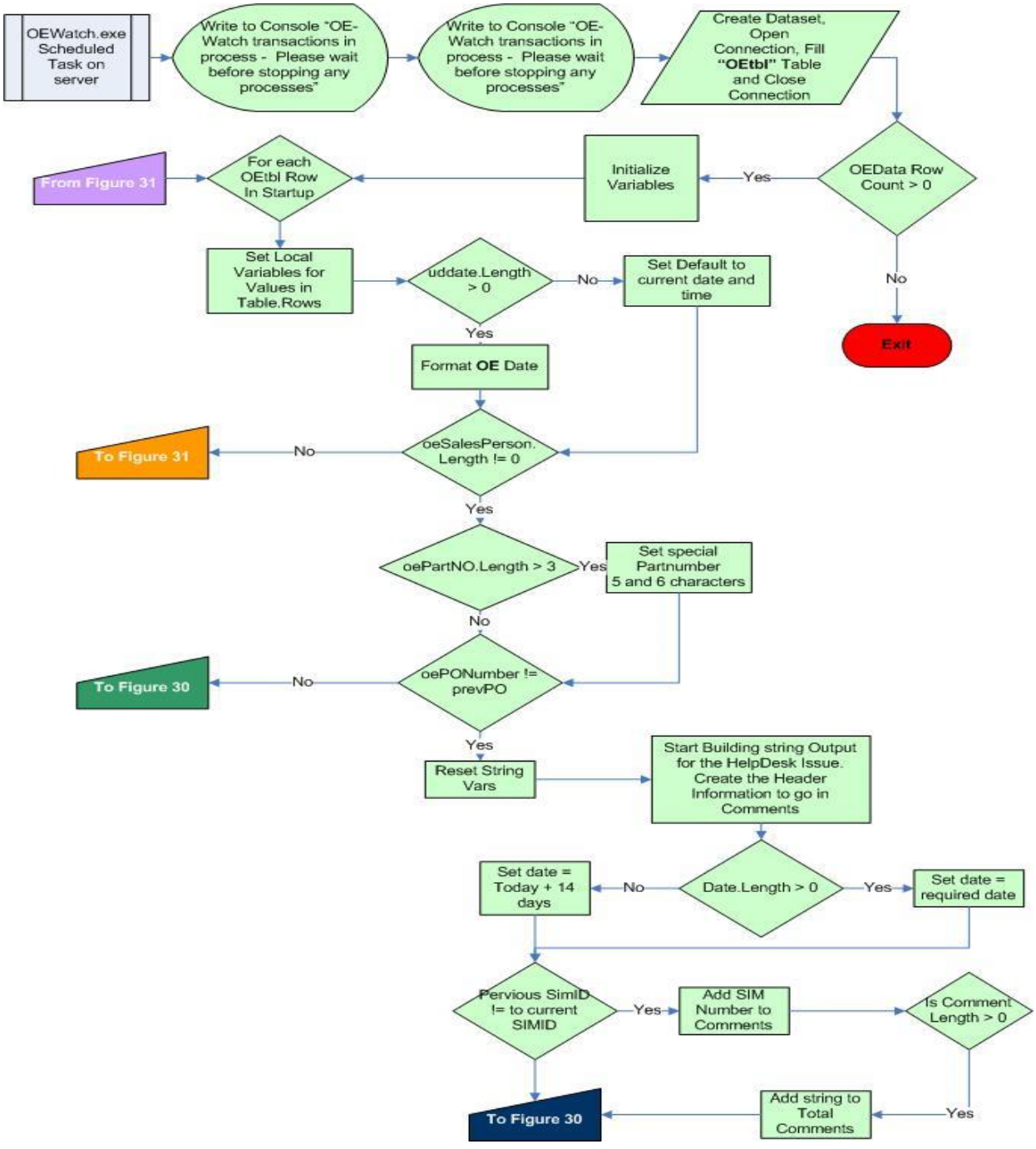

Figure 29 OEWatch – Logic Diagram 1 of 4

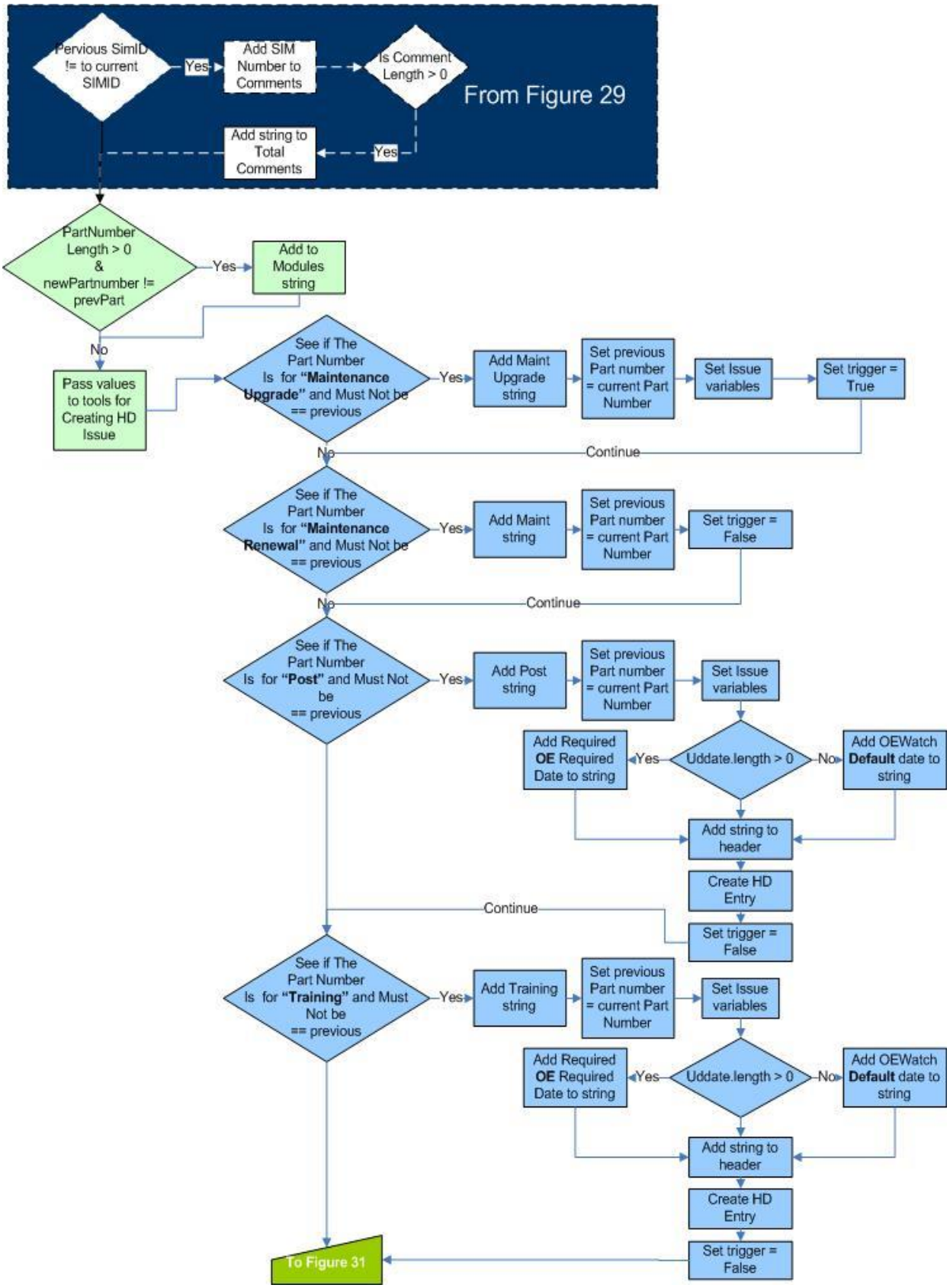

Figure 30 OEWatch – Logic Diagram 2 of 4

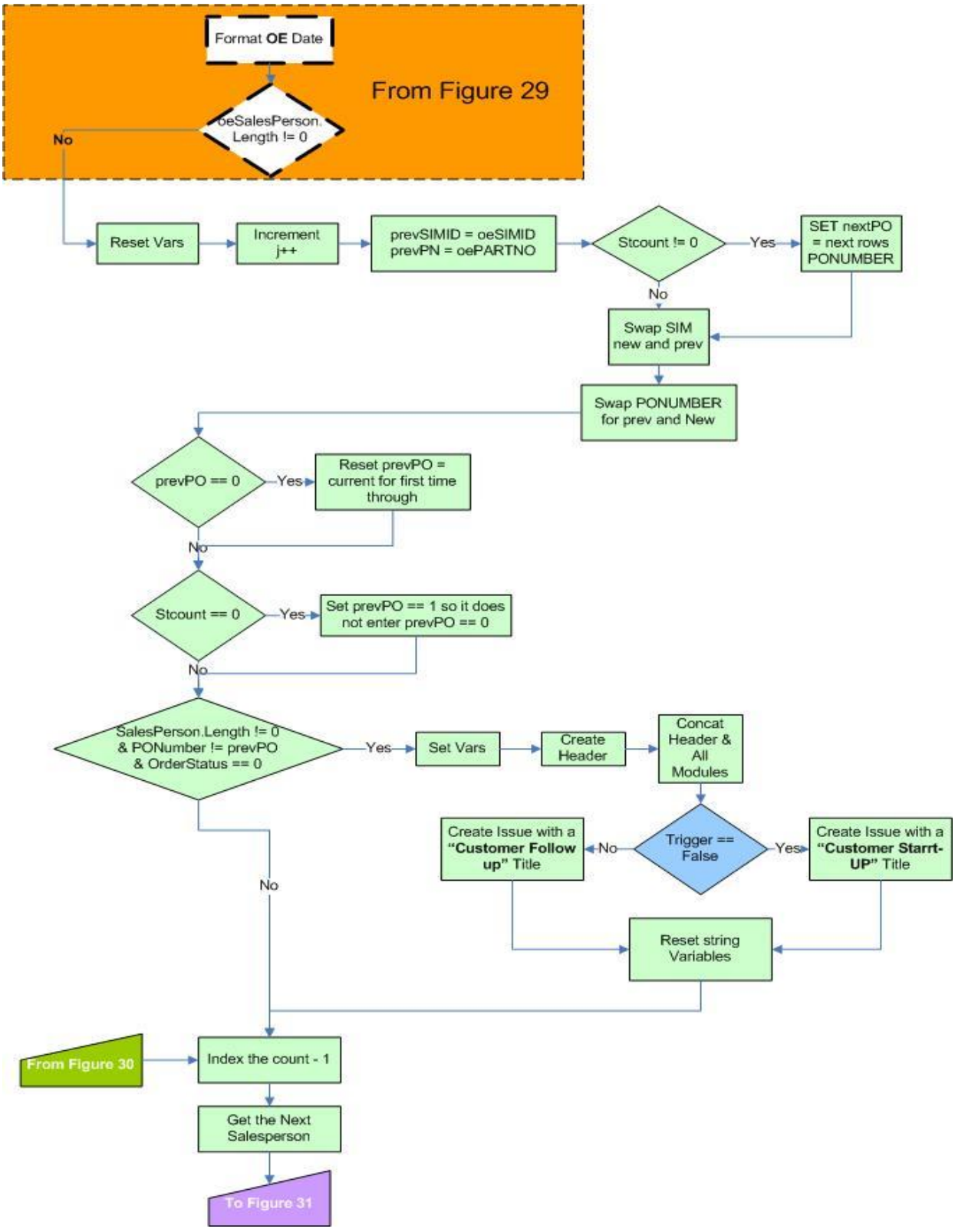

Figure 31 OEWatch – Logic Diagram 3 of 4

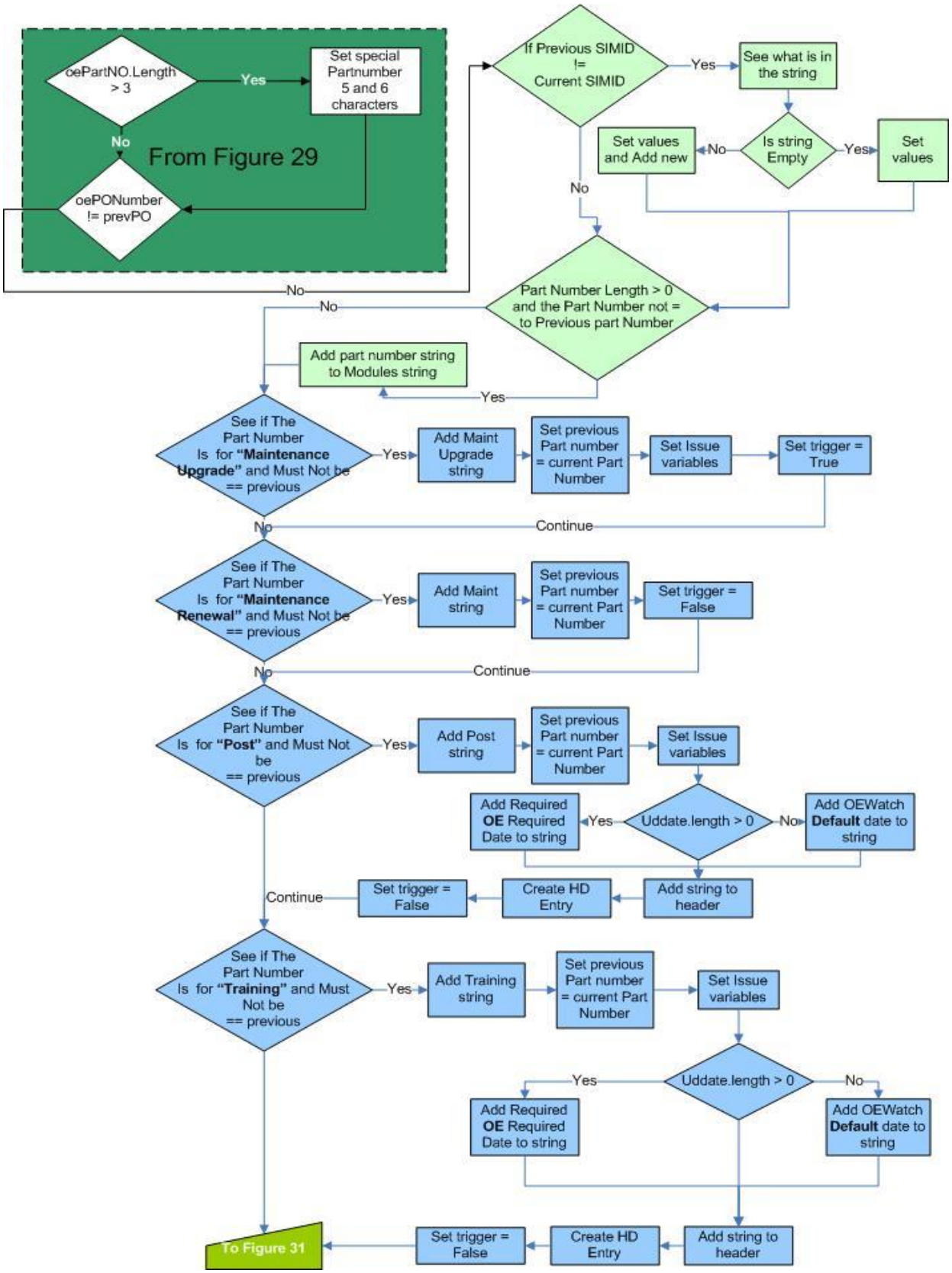

Figure 32 OEWatch – Logic Diagram 4 of 4

References

1. Borland.com, Borland InterBase Home Page. 12 December. 2005. <http://www.borland.com/us/products/interbase/index.html>

2. Boumphrey, Greer, Raggett, Raggett, Schnitzenbaumer, Wugofski. Beginning XHTML . Wrox Press Limited, April 2000.

3. csharphelp.com. "C# Help". 3 June. 2005. <http://www.csharphelp.com/>

4. Developer.com, "Issues and Challenges Facing Legacy Systems" ; Federico Zoufaly , http://www.developer.com/java/other/article.php/1492531, [15 Jan 2005]

5. devx.com. FreeVBcode, 2005. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/off2000/html/acconwhentoupsizemdbtosqlserver.asp>

6. Dewson, Robin. Beginning SQL Server 2000 Programming. Apress Publishing. 2001

7. Intuit Developer Network. QuickBooks SDKs. "Overview" 10 December. 2005. <http://developer.intuit.com/QuickBooksSDK/Briefing/?id=110>

8. MSDN. "When to upsize a Microsoft Access database to Microsoft SQL Server". 2002. < http://msdn.microsoft.com/library/default.asp?url=/library/en-us/off2000/html/acconwhentoupsizemdbtosqlserver.asp>

9. Oberg, Robert J. C# Using .NET. Prentice Hall PTR, 2001

10. Planet-Source-Code.com, Home Page. 9 August. 2005. <http://www.planet-source-code.com/ >

11. QODBC.com, "QODBC is the driver for accessing the data in *QuickBooks* Accounting Files". http://www.qodbc.com/qodbc.htm [March 2004]

12. Scheppa, David. Microsoft ADO.NET. Microsoft Publishing. 2002.

13. w3schools.com. "ASP Tutorial". Jan 18. 2005. <http://www.w3schools.com/asp/default.asp>

14. w3schools.com. "SQL Tutorial". Feb 20. 2005. <http://www.w3schools.com/sql/default.asp>

15. Delphi.about.com. "Introduction to InterBase SQL". 10 November. 2005. <http://delphi.about.com/b/a/007325.htm>