



**Improving Budget Management and Cash Flow
Forecasting for Projects**

submitted to
Karen Buelow
Dr. Roman Tankelevich

by
Kurtis Griess
Nicholas Henry

on
June 21, 2007

Table of Contents

ABSTRACT	1
1.0 INTRODUCTION TO THE PROJECT	2
2.0 REQUIREMENTS AND SPECIFICATIONS.....	3
2.1 SYSTEM REQUIREMENTS	3
2.2 PROJECT REQUIREMENTS.....	3
2.3 SPECIFICATIONS.....	4
3.0 APPROACH TO SOLUTION.....	5
3.1 MAIN PROBLEMS FACED	5
3.1.1 <i>Complex Payment Terms Problem</i>	5
3.1.2 <i>Collecting Data from QB and Intermediate Spreadsheets</i>	6
3.1.3 <i>Use Excel, Access, or MySQL for the Solution</i>	6
3.2 DISCOVERY OF SOLUTIONS AND ANALYSIS	6
3.2.1 <i>Solutions to Handling Complex Terms</i>	6
3.2.2 <i>Solutions for Collecting Data from QB and Intermediate Spreadsheets</i>	7
3.2.3 <i>Solutions for Using Excel, Access, or MySQL</i>	8
3.3 DOCUMENTATION AND PRESENTATION OF POSSIBLE SOLUTIONS TO THE CLIENT	9
3.4 SOLUTIONS CHOSEN BY THE CLIENT	10
3.4.1 <i>Chosen Solution to Handling Complex Terms</i>	10
3.4.2 <i>Chosen Solution for Collecting Data from QB and Intermediate Spreadsheets</i>	10
3.4.3 <i>Chosen Solution for Using Excel, Access, or MySQL</i>	11
4.0 DESIGN OF THE SOLUTION.....	11
4.1 HIGH-LEVEL DIAGRAMS OF THE SOLUTION	11
4.1.1 <i>Gathering Data</i>	11
4.1.2 <i>Putting it all Together (Gathering Data from QB and Spreadsheets)</i>	12
4.2 IN-DEPTH DESIGN OF THE SYSTEM.....	13
4.2.1 <i>Back-end: Data Storage and Table Relations</i>	13
4.2.2 <i>Front-end: User Interface and Interactions with the System</i>	13
4.3 IMPORTANT CHARACTERISTICS OF QUICKBOOKS CONSIDERED IN DESIGN	14
4.4 DELIVERABLE: “SYSTEM SPECIFICATIONS SPREADSHEET”	16
5.0 PRELIMINARY IMPLEMENTATION AND RESULTS	17
5.1 COST STATUS SUMMARY REPORT (CSSR.PHP)	17
6.0 SCOPE AND PROJECT PROGRESSION.....	19
7.0 CONCLUSIONS AND FUTURE DIRECTIONS.....	20
LESSONS LEARNED.....	20
<i>Kurtis Griess</i>	20
<i>Nicholas Henry</i>	21
GLOSSARY	22
APPENDIX A: ORIGINAL ANALYSIS DOCUMENTATION.....	26
APPENDIX B: IMPROVING BUDGET MANAGEMENT AND CASH FLOW FORECASTING FOR PROJECTS PRESENTATION	41
APPENDIX C: REQUESTED CSSR TEMPLATE	50
APPENDIX D: “SYSTEM SPECIFICATIONS SPREADSHEET”	51
APPENDIX E: CSSR CODE IN PHP	57

Abstract

Omnitech International, Inc designs, engineers, develops, and installs can manufacturing systems. Omnitech needs an improved budget management and cash flow forecasting system for each of their projects. Their accounting system, QuickBooks, handles accounting, but not cash flow or budgeting. Therefore, Omnitech handles cash flow and budgeting by a process requiring several users, several spreadsheets, and duplication of data entry. This system is confusing, time consuming, and duplicates efforts for parties in charge of keeping the system updated and accurate.

Omnitech wants to have cash flow and budget status reports automatically available each week (and on demand). Both reports must have drill-down capability so that management can view high-level data and other users may view more in-depth data. The reports must be user-friendly and the data entry process must be efficient and easy.

Our team analyzed possible solutions for the project, presented our solutions and recommendations to the client, and finally designed and began implementation of the client chosen solution.

1.0 Introduction to the Project

“Omnitech International, Inc. designs, engineers, develops, and installs two-piece drawn and ironed (D&I) can and can end manufacturing systems on a global basis.”[1]
Omnitech needs to improve their cash flow forecasting¹ and budget status² management system. Thus, Omnitech's Chief Financial Officer (CFO), Karen Buelow, graciously submitted the project to the Colorado School of Mines Mathematics and Computer Science Field Session Project List.

Upper Management at Omnitech has summary reports of the cash flow and budget status in spreadsheets. The process of updating and maintaining these reports requires several users to manually enter and re-enter data into several intermediate spreadsheets and Omnitech's accounting system, QuickBooks (QB). This process of updating spreadsheets is inefficient and results in inaccurate, outdated data.

To improve this system, Omnitech asked us to eliminate excess steps in the process and update the reports automatically with the most recent data possible. Upper Management must be able to view the reports in their desired layout and data-entry users must be accommodated with user-friendly forms to enter necessary data. To eliminate extra steps in the process, the system must extract data from QB instead of users re-entering data.

At the beginning of this project, we studied the requirements and spent time learning the current system: how each spreadsheet is used, how QuickBooks is used, where data comes from, and who the users are. We continued by identifying problems with the current system, discovering solutions to those problems, and analyzing each of the solutions. After the first week of analysis, it became clear to our team that the scope of this project was greater than the six weeks scheduled. Therefore, Omnitech agreed to keep our scope for this project flexible.

Originally, there were very few specifications, which left the overall solution to this project open-ended. Per client request, we analyzed each feasible solution and finally presented our recommendations to the client. After our presentation, Omnitech chose to use a MySQL database to store the cash flow and budget status data and PHP³ to create forms and reports accessible on the web. Under the revised scope, our final deliverable is a table, similar to a Use Case list, which will serve as an implementation guide and the foundation for a user manual.

¹ Cash flow forecasting: Calculating the future cash entering and cash leaving a business during a period of time

² Budget status: The comparison of actual revenues and costs to the approved budgeted revenues and costs for a project (we title this budget status the “Cost Status Summary Report”)

³ PHP (Hypertext Preprocessor): A server-side scripting language that processes user requests and allows programmers to create dynamic web pages

2.0 Requirements and Specifications

2.1 System Requirements

1. Automate cash flow forecast
 - 1.1. Must be able to take a snapshot of the cash flows
 - 1.2. Must display reports for financial data on every level (from a summary of all projects to specific details of a budget item)
 - 1.2.1. System will create a monthly (or weekly) cash flow report for each project
 - 1.3. Update the cash flows with actual data extracted from QuickBooks and other sources
 - 1.3.1. Extract data every week or on demand and put it into a database with other purchase orders and budget data
 - 1.4. Formulas and algorithms based on “rules” will automatically calculate cash flows
2. Automate project Cost Status Summary Report
 - 2.1. Must update project cost status summary every month or on demand
 - 2.2. Updated report will be used as purchase order review and cost review
 - 2.2.1. Project manager can review and verify whether POs are written and sent
 - 2.2.2. Report must separate and list categories and items in manner consistent with Item Master Database, QB, etc.
 - 2.2.2.1. Display manufacturing purchases on high level (not on item level)
 - 2.3. Automated updates for actuals from QB including purchase orders issued to date (PO number, issue date, and PO amount) and cash-out
 - 2.4. A snapshot view of the budget status and project end forecast of profit will be available
 - 2.5. Report displays details by project and by cost category (equipment, direct costs - travel, contractors, materials, shipping)
 - 2.6. Summary and “drill-down” functionality
 - 2.7. Original project budget data entry must look similar to current spreadsheets
3. System will link to QuickBooks, Item Master Database, and other systems that must remain separate from our team’s program.
4. System accepts changes to data that modify cash forecast
5. System will e-mail accounting to issue an invoice when appropriate
6. System must handle complex payment terms
 - 6.1. Easy to enter new terms in as few steps as possible for user
7. System must have capability for user-generated reports
8. System must be web-accessible
9. System must work with Item Master Database

2.2 Project Requirements

1. Help standardize Omnitech’s terms or vocabulary to make reporting possible between different systems; however, it is Omnitech’s responsibility to make the final decisions on terminology. For the project, the following will need to be updated to conform to the approved terminology:
 - 1.1. Project Budgets

- 1.2. Shipping Spreadsheets
- 1.3. QuickBooks's chart of accounts
- 1.4. Item Master top-level assemblies/assemblies
- 1.5. Possibly BigTime and some other spreadsheets
2. Present design to users to ensure format meets their needs. It is the users' responsibility to voice their needs for this system.
 - 2.1. Karen
 - 2.2. Dave
 - 2.3. Shelley
3. Match categories from QuickBooks, Purchase Order Spreadsheet, Dave's Budget Spreadsheet, and the Item Master Database
4. Create formulas and algorithms to update the cash flows
 - 4.1. Define rules on when cash comes in and flows out

2.3 Specifications

1. Cost Status Summary Report will have columns as in "KJOCS2 Cost Summary & Budget_03May07.xls"
2. Cost Status Summary Report will have rows as in "MX3 EQUIP DELIVERIES and VALUES_1May07.xls"

3.0 Approach to Solution

Our approach to the solution was a multi-step process. First, we gathered requirements (see Section 2.0) and reflected them back to the client to make sure we were on the right track. We continued to update and add clarifications to our requirements throughout the project. The next step was to identify the main problems facing us. Next, we discovered possible solutions to those problems and analyzed each. After we documented the analysis, we presented all the possible solutions to the client, gave our recommendation, and the client chose the best solutions.

Upon client request, we proceeded with further analysis of some solutions, started designing their chosen solutions, and did some implementation. Since the project scope was larger than anticipated, our client and our team agreed the deliverable should be a design of the client's chosen solution that will lead to complete implementation after our field session ends. Therefore, our approach to the solution does not include a full-scale implementation or testing of a product.

3.1 Main Problems Faced

We faced three main problems to automate the reports: (1) replacing manual calculations of cash flows based on complex terms with automatic calculations, (2) obtaining data from QuickBooks and intermediate spreadsheets and, (3) use either Microsoft (MS) Excel, MS Access, or MySQL (with PHP) for data storage and management for the reports.

3.1.1 Complex Payment Terms Problem

Payment terms are the manner and schedule in which a company pays cash for purchases. Payment terms that Omnitech uses for small purchases are generally simple (e.g. make full payment for the purchase 30 days after receiving the invoice), but for bigger purchases, the terms are more complex. Complex payment terms (or "complex terms") are payment terms that are broken up into several pieces with different percentages of the total payment to be made on specific trigger dates⁴ (e.g. pay a 30% down payment, 60% of the payment 120 days after delivery, and 10% of total after installation).

The first problem is that QuickBooks cannot accurately make calculations of cash flows in or out because it does not have functionality for complex terms. In other words, QuickBooks cannot calculate future cash-in or cash-out because it does not understand complex terms. Therefore, users at Omnitech have to make the calculations manually.

⁴ Trigger dates: Payment terms specify when to make payments based on dates of certain events (e.g. receipt of an invoice, shipment of purchased items, and installation of items). The dates of these events are trigger dates because they "trigger" cash being paid out at some time. The solution will make calculations for cash flow based upon estimated trigger dates

3.1.2 Collecting Data from QB and Intermediate Spreadsheets

The second problem is twofold. First, users do not have a convenient way of collecting necessary data as entered into QB. Second, users enter other data (e.g. ship dates of purchases) into various intermediate spreadsheets, which causes confusion and wastes time. We need to find a solution to get the data from QB and organize it in the way our client wants. We also need to consolidate data not from QB and ensure all users enter that data into and access the data from one “source of truth”⁵.

3.1.3 Use Excel, Access, or MySQL for the Solution

We need to store data for calculations and display it in reports. Excel has a convenient table format that is familiar to most professionals. Access has tables and is very useful for setting up databases, but many professionals are not as familiar with it as Excel. MySQL is a powerful database; however, it requires a programmer to create an interface for everyday users. Each program would suit the needs of the solution, but each has pros and cons.

3.2 Discovery of Solutions and Analysis

We discovered a number of possible solutions for each problem and analyzed each (see Appendix A for the original analysis of the problems and solutions). Our analysis began with detailed descriptions of the problems and descriptions of our possible solutions. Our descriptions of the solutions included high-level design to show how they would work (including diagrams showing databases, programs, and data flow). In order to do the analysis, we also completed small-scale implementations to test the feasibility of the solutions and discover other potential issues. Next, we analyzed the pros and cons of each of the solutions and finished with our recommendations.

3.2.1 Solutions to Handling Complex Terms

We found three solutions to handling complex terms (see 3.1.1) while reducing the manual data entry and calculations by the users. Below is a high-level description of each of the proposed solutions, but not all information is explained. Please refer to Appendix A: Original Analysis Documentation” for descriptions that are more complete, diagrams, pros and cons, and further analysis of the solutions.

Option 1: Add Complex Terms to Current Terms Field in QB

The first option requires users to select terms titles⁶ (complex terms included) in QB using a drop-down menu and our system would automatically extract the title from QB and make calculations based on the title and other data (trigger dates and percentages).

⁵ Source of truth: The data for calculations and displaying in the reports should come from one official location so its accuracy is not questioned

⁶ Terms title: Each complex term has a title given to it. E.g. 30% down payment, 60% of the payment 120 days after delivery, and 10% of total after installation could be titled and referred to as “30%-DP/60%-120d/10%-AI”

Extra data pertaining to the terms (e.g. number of days until trigger date and percentage of payment) needs to be properly associated to the terms title in our database for the calculations to take place. Therefore, users must occasionally enter the extra data into our solution.

Option 2: Create New Terms Field in QB

The second option is very similar to the first option, but requires users to enter the terms titles manually into QB every time. The only possible benefit for this is that users will not have to worry about looking through a long list of terms to select their particular title.

Option 3: Users Enter Data into Solution

The third option requires users to select the terms titles (including complex terms) in our solution. The users also enter the extra data (such as number of days until payment and percentage of payment) into the solution.

We recommended *Option 1: Add Complex Terms to Current Terms Field in QB* because it would be easiest for the users and users would continue to work in QuickBooks (which they are already familiar with). However, the final solution chosen by the client was a modification and combination of Option 1 and 3. Please refer to Appendix A for further explanation why we recommended Option 1 over the others.

3.2.2 Solutions for Collecting Data from QB and Intermediate Spreadsheets

We found two solutions each for collecting data from QB and the intermediate spreadsheets (see above, 3.1.2). First, we describe the possible solutions for collecting the data from QB and then describe possible solutions for collecting data from the spreadsheets. Please refer to Appendix A for more descriptions and analysis of the solutions.

Collecting Data from QB:

Option 1: Link Data from QB to Solution

The first option for collecting data from QB requires the use of the QuickBooks Open Database Connectivity⁷ (QODBC) to link data from QB to update data in the solution from QB automatically.

Option 2: Link Data between QB and Solution

The second option for collecting data from QB uses the QODBC to connect data in both directions: from QB to the solution and from the solution to QB. For this solution, we proposed to replace QB functionality by our system and require users to enter more data in our solution (in addition to payment terms data).

We originally preferred and recommended *Option 1: Link Data from QB to Solution* because we thought it would be easiest for the users and not as risky as Option 2. We

⁷Open Database Connectivity (ODBC): Provides a connection between a program and a database. The QODBC specifically provides a connection between QuickBooks and a database

thought it would be difficult to ensure data in our solution is sent correctly back to QB if we chose Option 2. However, during our presentation to the client, we gained further information that led us to believe that Option 2 would be the best choice. It would not be too difficult to implement correctly and it would better suit our client's needs.

Collecting Data from Intermediate Spreadsheets:

Option 1: Link Current Spreadsheets to Solution

The first option for collecting data from spreadsheets requires users to continue using their spreadsheets while our solution links to the data and then performs the necessary calculations.

Option 2: Copy Data from Spreadsheets into Solution

The second option for collecting data from spreadsheets requires all the important data from the spreadsheets to be copied to our database. From that point on, users of the intermediate spreadsheets would enter data into forms in our solution instead of spreadsheets.

We recommended *Option 2: Copy Data from Spreadsheets into Solution* because users would enter and manage data in one location, it would be easy for the users, and it is more likely to remain a reliable long-term solution.

3.2.3 Solutions for Using Excel, Access, or MySQL

Each of the proposed databases (Excel, Access, and MySQL) can perform the necessary functions needed to meet the requirements of this project. However, each has additional characteristics or functions that make them favorable or unfavorable. Below is a brief list of some of the characteristics and functions we took into account. Further description of each system and the pros and cons are in the sub-options of the "Gathering Data from QuickBooks" heading in Appendix A.

Excel:

- Users are familiar with Excel
- Users can manipulate formulas, results, and report formats by themselves
- Linked data is not real-time
- Little control over user access
- Only one person can modify spreadsheet at a time
- Manipulating spreadsheet cells can compromise solution integrity (can create new "source of truth")

Access:

- Easy for users
 - Access has built-in report generation/saving capabilities
 - Users have more control over the data they view
- No programmer needed
- Users may want to work with data themselves
 - Users will have to export data to Excel

MySQL with PHP:

- Easy for user
- Works well with other systems Omnitech currently uses
- No need for VPN when working remotely
- Maintenance requires programmer
- Users may want to work with data themselves
 - Users will have to export data to Excel

We recommended using either Access or MySQL primarily because they offer more security (e.g. different levels of user access protected by user names and passwords) and they are more likely to keep the system reliable in the long-term.

3.3 Documentation and Presentation of Possible Solutions to the Client

We submitted a document of our analysis to our client (Karen Buelow, CFO) for review. Our document was originally very technical, so after our client reviewed the document, she requested that we gear this document towards the users who would not understand the technical descriptions. Therefore, we reduced the technical descriptions in the paper and presented the pros and cons from the standpoint of the user. After we changed the paper, our client told us to focus on creating a presentation, which would be more convenient for the users to review.

We created the presentation of our analysis in a similar format as the document: with a description of the problems, description of the solutions, pros and cons, and recommendation. We spent many days working on the presentation to make sure that the users would be able to understand our solutions and that our analysis was complete. We sent our presentation to our client to review and altered the presentation as requested. Our presentation includes more detailed diagrams and analysis applicable to the users than the written document.

We presented our solutions on June 5 to the following audience:

- Karen Buelow, CFO
- Dave Harrison, V.P./General Manager
- Jim Weed, Manufacturing Manager
- Shelley Walton, Controller

We presented the solution again on June 8 to the following:

- Mike Cistone, Commercial Manager – Procurement and Logistics
- Roy Brace, Procurement Specialist
- Jodi Frisch, Admin and Accounting Assistant

Please find the presentation in Appendix B. It shows all the descriptions and analysis in the first 45 slides. After our first presentation, we added the names of the attendees of and the outcomes to the end of the slideshow. We emailed the presentation to potential users of the system who could not attend the presentations.

Ultimately, the attendees of the first presentation and our client made the decision on the solutions to implement. They were able to eliminate some of the possibilities immediately after the presentation, but they asked for more analysis on certain solutions. For example, they asked us to explore the reporting capabilities of MySQL using PHP or other reporting software (specifically Crystal Reports). After this further analysis, Karen made the final decision on which solution to implement. At that point, we had the main specifications of the system and we were able to continue design.

3.4 Solutions Chosen by the Client

3.4.1 Chosen Solution to Handling Complex Terms

For handling complex terms (see 3.1.1 and 3.2.1), Omnitech chose to implement a variation of *Option 1: Add Complex Terms to Current Terms Field in QB* and *Option 3: Users Enter Data into Solution*. Users will enter purchase orders⁸ (POs) in QB and enter the terms titles in the purchase order by selecting a title from a drop-down menu. If the terms title is not yet in the drop-down menu, the user must enter the terms title and the extra data (“trigger dates” and percentages) using a form created for the solution. The solution will then send the terms title back to QB and make the terms title available in the drop-down menu for future use.

3.4.2 Chosen Solution for Collecting Data from QB and Intermediate Spreadsheets

Collecting Data from QB

The client selected *Option 2: Link data between solution and QB* as their solution for obtaining data from QB. Using the driver for QODBC, our solution will connect to updated data in QB. Our solution will connect back to QB in order to send the new terms titles that users enter when they cannot find their desired terms title in the drop-down in QB. This solution requires the purchase of a \$500 dollar QODBC driver, which Omnitech took into account before selecting this option.

Collecting Data from Intermediate Spreadsheets

The client chose *Option 2: Copy Data from Spreadsheets into Solution* as their solution for obtaining data from the intermediate spreadsheets. We will copy the data from the spreadsheets into the solution's database and all data not entered into QB (e.g. ship dates) will be entered directly into the solution in the future using forms.

⁸ Purchase order: Order form to make a purchase from a vendor

3.4.3 Chosen Solution for Using Excel, Access, or MySQL

Omnitech chose to use a MySQL database to store and manage the data, and then use PHP to display queries to the web. Implementation will require setting up the database with the appropriate fields and populating the database using links to QB and data obtained from forms. Before implementation, some design must take place to ensure the database is set up in the best manner.

4.0 Design of the Solution

4.1 High-level Diagrams of the Solution

4.1.1 Gathering Data

Our solution must gather data from QuickBooks. To accomplish this, our solution will use an ODBC driver to create a connection between the QuickBooks database and our system. Users will enter payment terms data and data previously entered into the intermediate spreadsheets into our solution. Our system will automatically update QuickBooks with any data it needs (especially terms titles) after each transaction (See Figure 1).

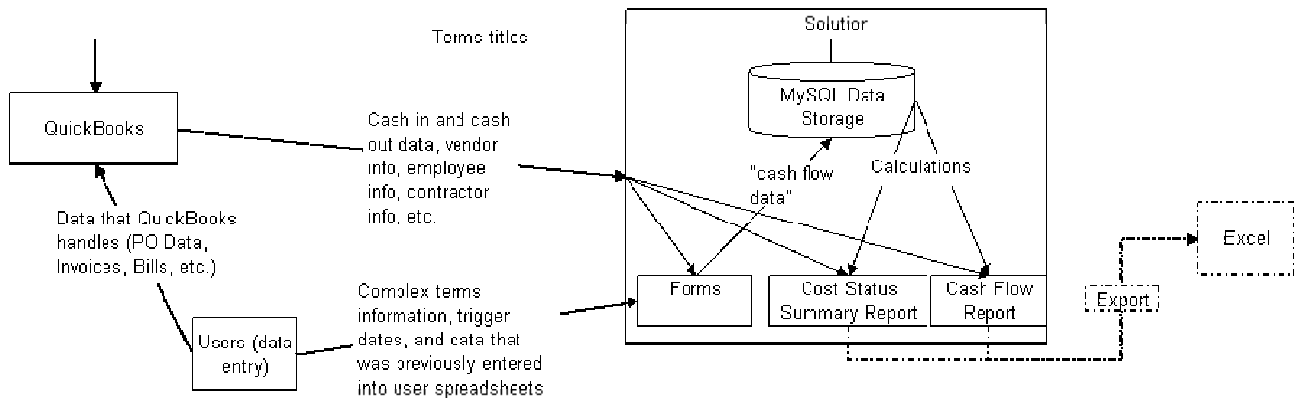


Figure 1: Users will use our system to enter and store new complex terms titles and trigger dates, and update data previously entered into the intermediate spreadsheets

Our solution must gather data that users previously entered into their spreadsheets. Our solution will provide several forms for users to enter their data and send that data to our database (instead of their spreadsheets). This will be more efficient and easier for users than maintaining their current spreadsheets. Once the system is set up, we will copy the data from the users' spreadsheets to our system and from then on, all users will enter all their data into the system through forms (See Figure 2).

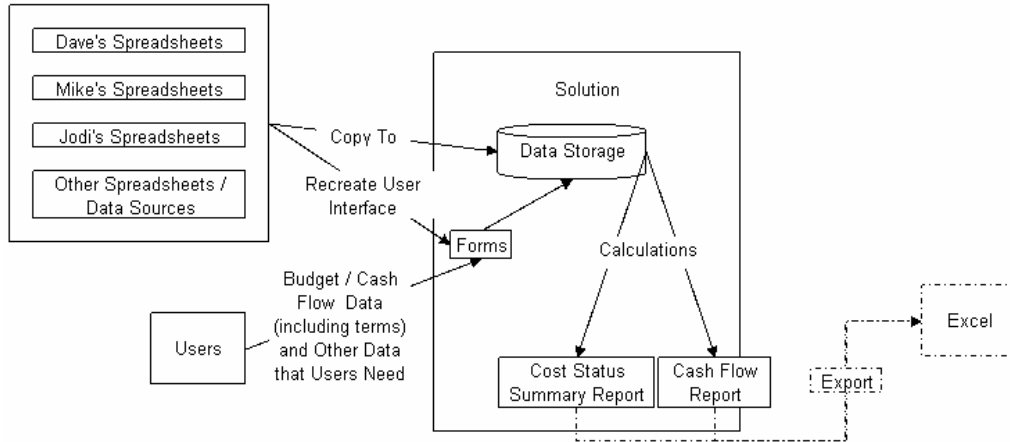


Figure 2: The system will have forms and reports for users to enter and view data. A programmer will move the data from the users' spreadsheets into our system. From then on, users will enter cash flow and budget related data into our system. The users will then be able to create reports with our system and export to Excel if they want to do further analysis.

4.1.2 Putting it all Together (Gathering Data from QB and Spreadsheets)

Figure 3 shows a diagram of the data flow, main actors, and the programs the completed solution will use. It combines the diagrams in Figures 1 and 2. Please see the caption for a description of the figure.

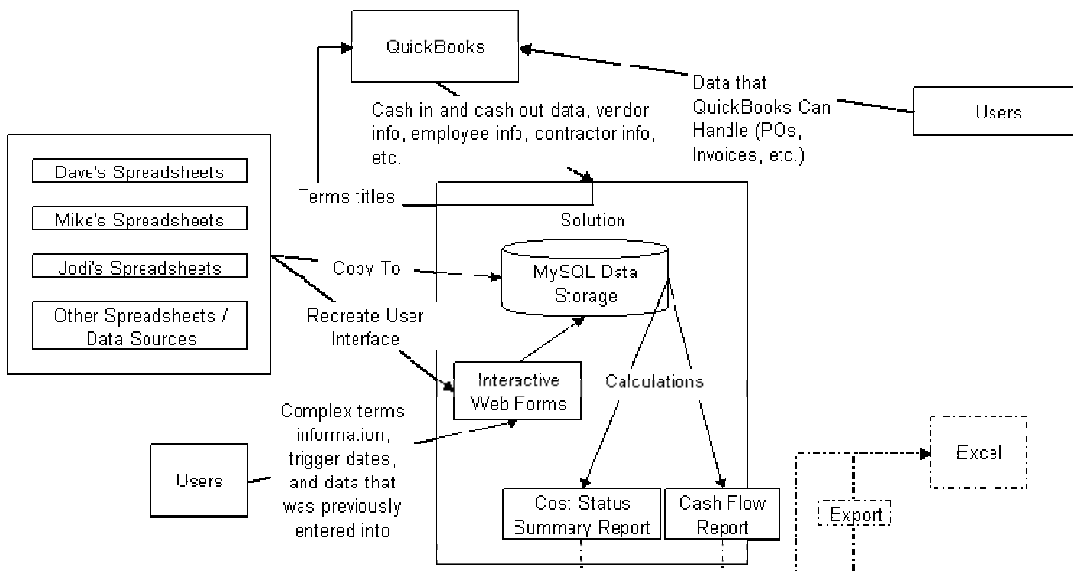


Figure 3: For the final product, users will enter data previously entered in to spreadsheets into our system. They will also enter terms titles and trigger dates into the solution and make updates to trigger dates in the solution, but will continue to use QuickBooks for their usual activities. The system will provide users with easy-to-use forms and reports, as well as the ability to extract data from the system to Excel if users need to do further analysis.

4.2 In-depth Design of the System

4.2.1 Back-end: Data Storage and Table Relations

When completed, our system will rely on a relational MySQL Database to store complex terms and other data that QuickBooks does not handle. A relational database is a database, which contains tables for separate types of data. For example, the MySQL database will have one table for payment terms and another table for additional purchase order data. In non-relational databases, there will only be one table containing data for those separate types. The tables in a relational database can interact when they have related data by using a “primary key”. A primary key is a field that has all unique data that identifies a specific record. The tables interact when the primary key from one table is also in the other table (it is a “foreign key” of the other table). The table with the foreign key can associate its data with the other table’s data.

We will link the current QuickBooks data to our database by storing primary keys from QuickBooks such as ListIDs, TxnIDs, etc. in our system. Please see Figure 4 for a diagram showing the relations between databases containing payment terms data. OIITerms, OIITermItems, and TermsDates are all separate tables, but are related by their keys (e.g. ListID is the primary key for OIITerms and OIITermItems can associate its data with the OIITerms data by using the ListID key).

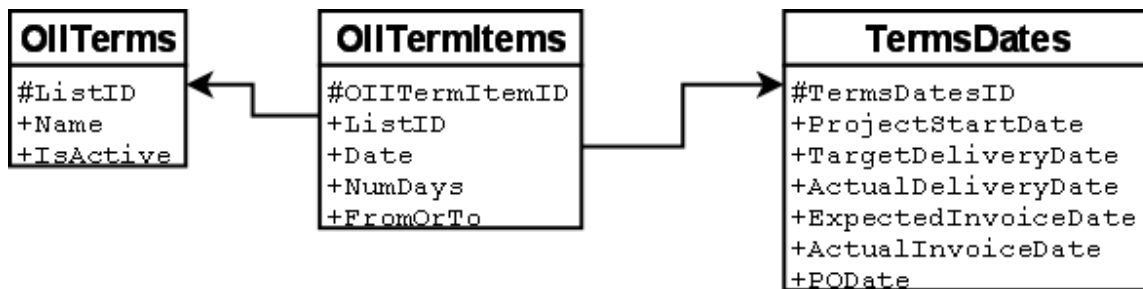


Figure 4: UML diagram for tables needed to handle complex terms.

4.2.2 Front-end: User Interface and Interactions with the System

Our system uses a web-based interface for gathering information from the MySQL and QuickBooks Databases and from Omnitech employees.

Forms for Data Entry and User Interaction

Forms will provide users a quick and easy way to input data. Wherever possible, our system will help users by automatically filling in fields in the forms with up-to-date data. The forms will also provide users with lists (drop-down menus) to help choose valid selections for certain fields. Eventually our system will use JavaScript to ensure that the data users have entered is correct and alert them if it sees any potential problems. When users submit forms, the system will confirm that the data was accepted or provide a detailed description of why the system was unable to submit the form.

Predefined and Custom Reports

The programmers will create predefined reports (according to client specifications) and create forms (or Wizards) to aid users in creating custom reports. During the creation of the predefined reports, the client will review the formats and content to ensure it will meet their needs. For some reports, such as the CSSR, the client has already expressed their desired format for the data (See Appendix C for the requested template for the CSSR). To create custom reports, users will choose the data fields they want and the system will use PHP to gather and organize the data. The predefined and custom reports will allow users to “drill-down”⁹ and view more detailed information on certain items.

4.3 Important Characteristics of QuickBooks Considered in Design

The most important characteristic of QB to consider is that QuickBooks uses flat file databases rather than a relational database. A flat file database is one table that contains all the data for the entire database across all different types of data. QB uses many flat file databases that contain all the data for different types instead of relating separate tables (containing data of different types that should be separate) using keys. Figure 5 shows some of the databases in QB.

Each list shown in the figure is a flat file database that contains many types of data and each list repeats much of the same data. A relational database instead would only have one database with separate tables that do not repeat data, but can link related data using keys. One important implication of QB’s flat file database is that the QODBC must update every database in QB when users enter updated data (such as new terms titles) into our system. In a relational database, the QODBC only has to update the data in one table.

⁹ Drill-down: View a high-level description in more detail. E.g. Drill-down to examine the cost of nuts and bolts that make up a larger machine.

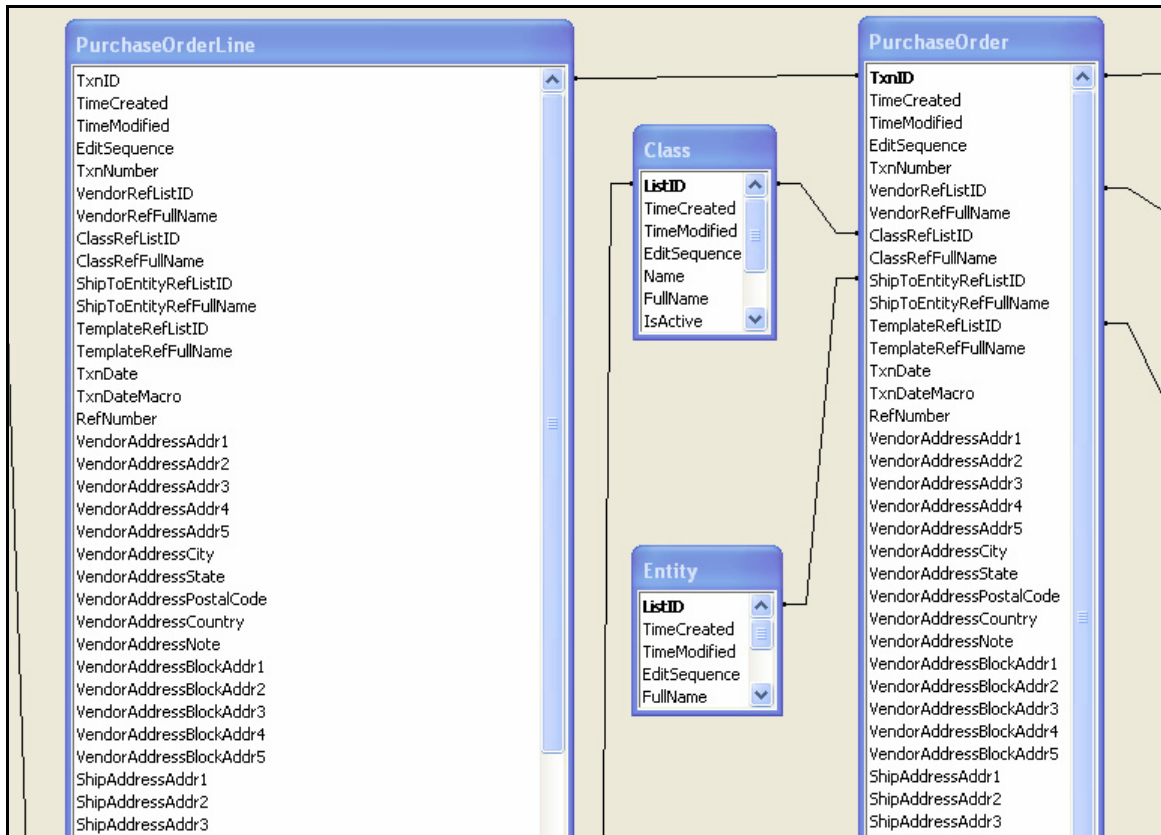


Figure 5: The PurchaseOrderLine Database contains all of the data that the PurchaseOrder Database already has. This is because the QuickBooks Database is a flat file database. (Lines are drawn between databases to show relationships even though the QuickBooks Database does not know of these relationships)

Due to the nature of flat file databases, QB lists more data than necessary in its databases (it repeats much of its data in each database). For example, the Cost Status Summary Report needs purchase order data. All the data from the purchase orders is in QuickBooks' PurchaseOrderLine database, but QB's PurchaseOrder and Terms databases repeat the same data (See Figure 6). Therefore, our design takes these multiple sources of the same data into account and the system will only collect data from one of the sources, not all of them. Not all of QB's flat file databases or the data they contain are required or incorporated into our design, but the QODBC website displays the entire schema¹⁰.

¹⁰ For a detailed list of all of the tables and fields in the QuickBooks database, please visit: http://www.qodbc.com/docs/html/qodbc/20/tables/table_info_all_us.asp

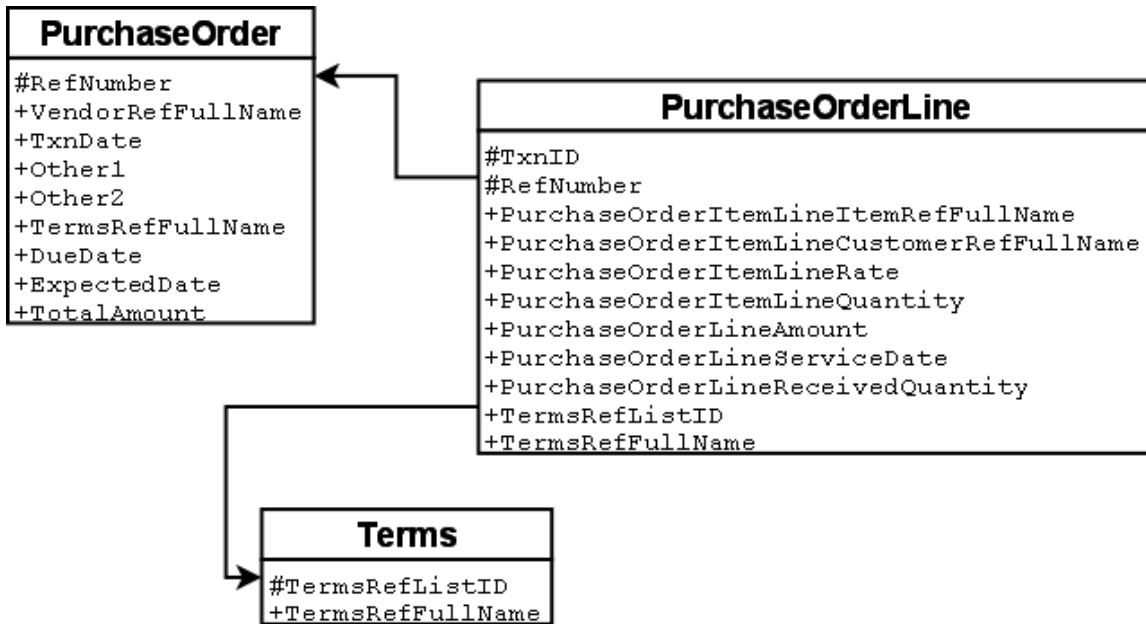


Figure 6: Simplified UML diagram for the QuickBooks database

4.4 Deliverable: “System Specifications Spreadsheet”

We organized specifications, requirements, and design of the system in a spreadsheet, which the client agreed would serve as the deliverable for this project. The tables from the spreadsheet are in this report in Appendix D. The spreadsheet shows the process of setting up and going through a project. For each stage of the process, our spreadsheet gives detailed information about necessary system functionality, required data, and the source of the data. It is in a format that will later help create a user manual and it provides the foundation for implementation.

5.0 Preliminary Implementation and Results

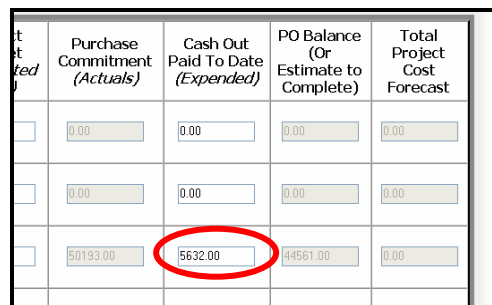
After our client chose to use a web interface with MySQL as a back-end¹¹, we began implementing the Cost Status Summary Report for Omnitech. We chose to implement parts of this report instead of the Cash Flow Report because although it is only a small portion of the final product, it was relatively straightforward and it allowed us to demonstrate solutions to all three problems¹² that Omnitech needed resolved. Due to the time constraints, implementation of the entire system was not completed.

5.1 Cost Status Summary Report (CSSR.php)

There were four goals for the Cost Status Summary Report. First, we wanted to prove that our system could extract data from QuickBooks and display it in a manner that would meet our client's needs. Second, we wanted to show that our system could generate predefined reports for Omnitech Employees. Third, we wanted to demonstrate that our system could perform calculations automatically when users entered new information. Finally, we wanted to illustrate that our system would be convenient and easy for Omnitech employees to use. The working code for the CSSR is in Appendix E.

To complete the first goal, getting the Cost Status Summary Report to extract data from QuickBooks, we had to complete several tasks. We began by setting up a web server that could connect to QuickBooks using a QODOBC DCOM¹³ driver. This included downloading and installing the new driver and giving internet accounts proper access to folders on the server. Once everything was set up and installed, we tested our connection to QuickBooks by using a simple PHP script that selected the names of the first fifty customers from QuickBooks. After a little tweaking, this was up and running and we began writing a query that would extract the data needed for the Cost Status Summary Report. Once the query was finished, we began formatting our raw data so that it would start to look similar to the Project Manager's current Cost Status Summary Report. Finally, after some fine-tuning, we were able to display the data that we extracted from QuickBooks in a manner that would meet our client's needs.

Our second goal was to create an easy way for Omnitech employees to generate their own Cost Status Summary Reports. We wanted employees to be able to choose a project and a beginning and end date and have the system produce a report that had only the data that they wanted. To accomplish this, we added a form at the top of the report where users can choose a project from a drop-down menu and then enter the dates they



	Purchase Commitment (Actuals)	Cash Out Paid To Date (Expended)	PO Balance (Or Estimate to Complete)	Total Project Cost Forecast
<input type="checkbox"/>	0.00	0.00	0.00	0.00
<input type="checkbox"/>	0.00	0.00	0.00	0.00
<input type="checkbox"/>	50193.00	5632.00	44561.00	0.00

Figure 7: When the user moves from the “Cash-Out Paid to Date” Field, the “PO Balance” field is calculated automatically

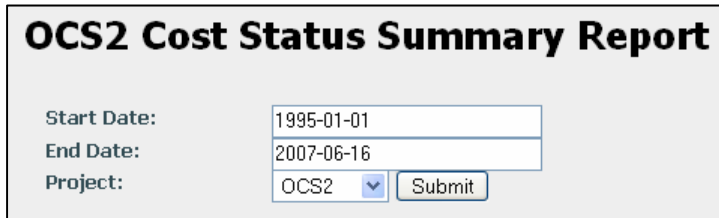
¹¹ Back-end: The internal workings of a system, including, but not limited to: data storage, formulas, and data relations

¹² Handling complex terms, gathering data from QuickBooks and intermediate spreadsheets, and which program to use for data storage and management (we chose MySQL with PHP)

¹³ Distributed Component Object Model (DCOM): Allows software components from several networked computers to communicate with each other

want into text boxes. Once the user presses submit, the page will reload with the data they want. The code for the form is in Appendix E.

The third goal was to show that the system could perform calculations automatically when users entered new information. To prove that this was possible, we added an event listener to the “Cash-Out Paid To Date” field that tells the system to automatically update the “PO Balance Field” with a new total when a user leaves the “Cash-Out Paid To Date” field.



OCS2 Cost Status Summary Report

Start Date: 1995-01-01

End Date: 2007-06-16

Project: OCS2

Figure 8: Form where users can choose a start date, an end date, and a project

format (YYYY-MM-DD) in the “Ending Date” field. We also fill the Start Date text box with a default date.

For the CSSR, we added some styles and formatting using Cascading Style Sheets¹⁴ to make it more engaging for users. We programmed the webpage to load and display the report piece-by-piece on the page at a time, rather than loading the entire report at once and then displaying. If the report is loaded all at once, it takes longer and we want the users to be able to see the report is loading sooner than later. This gives the user the impression that the system is instantaneous when it actually takes just as long for the reports to load completely.

We implemented several features to make the system easier for users. First, we programmed a form, which users will use to create predefined reports for projects over a chosen period. The form automatically fills the current date in the correct

¹⁴ Cascading Style Sheets (CSS): The language used to describe the presentation or style of an html document

6.0 Scope and Project Progression

We realized after three weeks of Field Session that the original scope for this project was too much for our team to complete in six weeks. Therefore, our client kept the scope for this project flexible until we approached week six and decided on the ultimate deliverables.

Gathered Requirements

Although this was an ongoing process, we spent a good part of the first week gathering requirements and specifications from our client and reflecting them back for review. In addition, we spent some time during the first week identifying Omnitech's main problems with their current system and understanding exactly what they needed our system to do.

Analysis

Once we discovered the main problems we faced, we spent a couple of days finding possible solutions. It took another two and a half weeks to finish small-scale design and implementation, to analyze the pros and cons, and to present our recommendations to the client.

Presentation of Possible Solutions

After we felt confident with our analysis of the possible solutions, we presented our findings to our client. The client made decisions on which solutions to implement. The client also requested some further analysis on the reporting capabilities of MySQL and this took another two days.

Design and "System Specifications Spreadsheet"

Our team spent the last two weeks working on the deliverable: the "System Specifications Spreadsheet" and other design. We met with our client to discuss and review the content of the spreadsheet.

Implementation of the Cost Status Summary Report

After our client decided to use a MySQL database with a web interface, we began implementing the Cost Status Summary Report. We knew we would not have enough time to implement the entire system, but we wanted to have a program to show the client. The implementation took a little less than a week.

7.0 Conclusions and Future Directions

Six weeks was not enough time for our team to complete this project for Omnitech; however, we have done all of the groundwork for what will be an invaluable asset to their organization. We were able to complete an in-depth analysis of possible solutions and helped our client come to a final decision on the system to implement. We also created the “System Specifications Spreadsheet,” which Omnitech will use to implement this system.

We hope that with our help, Omnitech will be able to start using the system for their next project, MX0003.

Lessons Learned

Kurtis Griess

- Presentations
 - Organizing presentations and documents when there are several options and sub-options is difficult and should be made as clear as possible
 - When summarizing presentations, tell who attended to give the results credibility and make its importance better known
 - Know your audience
 - Present the information in a way that is convenient and understandable to the client (audience)
 - If the presentation might be viewed later (outside of the verbal presentation), it should have sufficient details in the slides for readers to understand the big picture
 - Diagrams ("Big animal pictures") are extremely useful for planning and presenting
 - They help to see what connections there are, what data is exchanged and used, and the flow of the system
 - It is easier to show to users and explain difficult processes
- The difference between specifications and requirements
 - Specifications are more detailed solutions for the requirements
- Client relations
 - Show the client your work and keep them up-to-date on what you are doing
 - They help catch mistakes
 - They give valuable feedback
 - Everyone needs to be on the same page
 - Having a champion client who will push for progress in the project
 - They help get their co-workers involved
- Project processes
 - Gather (and revisit) requirements

- Create project timeline and define responsibilities
 - Analyze possible solutions
 - Present analysis and recommendations to client
 - Design
 - Implementation
- Presenting options the client and letting them choose the solution is very important because it gets their "skin in the game". In other words, it becomes their solution, not ours. This helps the client better understand the need for the system and how the system will affect them

Nicholas Henry

- Clients may not always have a clear-cut project in mind. If possible, it is beneficial to sit down with your client and have them show you exactly what they are doing now and why it does not work for them. You then not only will have a better idea of what the client wants (and does not want), but you also can see where some of the resources are that you will need.
- Find out your client's priorities. You may perceive a certain task as a high priority, when in reality your client just thinks it would be nice to have done someday.
- Think outside the box. Sometimes you can save yourself a lot of time, just by spending a little more time considering other options.
- Consider your audience. Most users do not want to know technical details or make technical decisions; usually they would only like to know how a choice would affect them (cost, time, ease of use, etc.)
- Even if you do not believe that a solution is probable, it is still important to present it to the client. He or she may have more reasons to choose one option over the others, or maybe they would like to combine several of them to make one that really works for them.
- Terminology is important. Adjusting some of the vocabulary in your presentations to reflect your client's needs can really help your client understand what you are trying to tell them.

Glossary

Access: Database Management System (DBMS) that functions in the Windows environment and allows you to create and process data in a database

Accounting Assistant: Employee who assists in Accounting and handles Accounts Payable

Actuals: Data extracted from the “source of truth”; actual data for costs and revenue (usually for comparing against budgeted amounts)

Back-end: The internal workings of a system; including, but not limited to: data storage, formulas, and data relations

BigTime: Omnitech’s time tracking software

Budget Management: The process of reviewing costs and purchase orders (at Omnitech, this will eventually include reviewing revenue)

Budget Status: The comparison of actual revenues and costs to the approved budgeted revenues and costs for a project (see Cost Status Summary Report)

Cascading Style Sheets (CSS): The language used to describe the presentation or style of an html document

Cash Flow Forecasting: Calculating the future cash entering and cash leaving a business during a period of time

Chief Financial Officer (CFO): The corporate officer primarily responsible for managing the financial risks of the business or agency

Chart of Accounts (QB Chart of Accounts): Different billing categories (Cost of Goods Sold, Income, Equity, Fixed Asset, etc.)

Commercial Manager: Employee in charge of making purchases and coordinating the shipping of equipment

Complex Payment Terms (Complex Terms): Complex payment terms are payment terms that are broken up into several pieces with different percentages of the total payment to be made on specific trigger dates (see Payment Terms, Trigger Dates)

Controller: Employee who manages the day-to-day accounting and banking operations

Cost Status Summary Report (CSSR): The comparison of actual revenues and costs to the approved budgeted revenues and costs for a project

Crystal Reports: Reporting software that is used to format, group, summarize and arrange data into a desired report format

Database: An organized collection of related data

Distributed Component Object Model (DCOM): Allows software components from several networked computers to communicate with each other

Drawn and Ironed: Process for making two piece cans in which a circular blank is drawn through a die to form a cup and then thinned to final dimensions by being forced through a series of two or more progressively smaller diameter ironing rings.

Drill-down: Gives the ability to view a high-level description in more detail. E.g. Drill-down to examine the cost of nuts and bolts that make up a larger machine.

Driver: A program that determines how a computer will communicate with a peripheral device

Drop-down: An item on a web page where users can click a button and choose from a list of acceptable options

Event Listener: An object in a program that waits for a certain action to occur before it performs an action in response

Excel: A spreadsheet application that creates spreadsheets, graphs, and does some basic sorting (see Spreadsheet)

Field: A part of a record used for a particular category of data

Flat File Database: A database where one table contains all the data for the entire database

Foreign Key: A column in a table used as a link to matching columns in other tables

Forms: An HTML document that presents the user with a series of interactive inputs

Front-end: Everything that the user sees and experiences on a web page

General Manager: Employee who oversees all aspects of projects and is responsible for profit/loss, budgets, performance, client relations, and project management

Interface: The interaction between the computer and the user

Invoice: An itemized list of goods shipped to a buyer, stating quantities, prices, shipping charges, etc.

Item Master Database: A program that Omnitech uses to track items (documents, parts, assemblies, etc.) that the company has used

JavaScript: A client-side scripting language for web pages

Microsoft (MS): The largest vendor of personal computer software applications and operating systems

MySQL: An open source relational database management system that uses Structured Query Language (SQL)

Open Database Connectivity (ODBC): Provides a connection between a program and a database. The QODBC specifically provides a connection between QuickBooks and a database

Payment Terms: The manner and schedule in which a company pays cash for purchases

PHP (Hypertext Preprocessor): A server-side scripting language that processes user requests and allows programmers to create dynamic web pages

Primary Key: A column in a table whose values uniquely identify the rows in the table

Procurement Specialist: Employee who handles purchasing of parts for manufacturing Omnitech equipment in-house

Purchase Order (PO): Order form to make a purchase from a vendor

QODBC: Provides a connection between QuickBooks and a database [see also Open Database Connectivity (ODBC)]

Query: An object that requests information from a database and creates a dataset of the requested information

QuickBooks (QB): The most commonly used small-business accounting and management software in the US

Real-time: The immediate availability of current data to an information system as a transaction or event occurs

Relational Database: A database that stores data in a structure consisting of one or more tables of rows and columns, which may be interconnected

Schema: A conceptual model of the structure of a database that defines the data contents and relationships

Manufacturing Manager: Employee who supervises the manufacturing and assembly of equipment built by Omnitech

Source of truth: One official location of data for calculations and displaying in the reports so its accuracy is not questioned

Spreadsheet: A screen-oriented interactive program enabling a user to lay out data on the screen

Term Titles: Each complex terms has a title given to it

Trigger Dates: Payment terms specify when to make payments based on dates of certain events (e.g. receipt of an invoice, shipment of purchased items, and installation of items). The dates of these events are trigger dates because they “trigger” cash being paid out at some time

UML Diagram (Universal Modeling Language Diagram): A general-purpose modeling language that includes a graphical notation used to create an abstract model of a system

Use Case: A complete sequence of related actions initiated by an actor; it represents a specific way to use the system

Virtual Private Network (VPN): Refers to a network in which some of the parts are connected using the public Internet, but the data sent across the Internet is encrypted, so the entire network is "virtually" private

Wizard: An interactive computer program acting as an interface to lead a user through a complex task using dialog steps

Acronyms

CFO – Chief Financial Officer

CSS – Cascading Style Sheets

DB – Database

DCOM – Distributed Component Object Model

PO – Purchase order

QODBC – QuickBooks Open Database Connectivity

QB – QuickBooks

MS – Microsoft

UML – Universal Modeling Language

VPN – Virtual Private Network

Appendix A: Original Analysis Documentation

Omnitech International, Inc Cash Forecasting



submitted to
Dr. Roman Tankelevich

by
Kurtis Griess
Nicholas Henry

on
May 29, 2007

Table of Contents

PROBLEM STATEMENT: OBTAINING DATA FOR CASH FLOW FORECASTING.....	1
PROBLEM: HANDLING COMPLEX TERMS	1
ANALYSIS AND HIGH LEVEL DESIGN OF POSSIBLE SOLUTIONS.....	1
<i>Option 1: Add Complex Terms to Current Terms Field in QB</i>	<i>1</i>
<i>Option 2: Create New Terms Field in QB</i>	<i>3</i>
<i>Option 3: Users Enter Data into Solution.....</i>	<i>4</i>
RECOMMENDATION FOR HANDLING COMPLEX TERMS.....	5
PROBLEM: GATHERING DATA FROM QUICKBOOKS.....	5
ANALYSIS AND HIGH LEVEL DESIGN OF POSSIBLE SOLUTIONS.....	5
<i>Option 1: Link Data from QB to Solution.....</i>	<i>Error! Bookmark not defined.</i>
<i>Option 2: Link Data between QB and Solution.....</i>	<i>8</i>
RECOMMENDATION FOR GATHERING DATA FROM QUICKBOOKS	10
PROBLEM: GATHERING DATA FROM SPREADSHEETS	10
ANALYSIS AND HIGH LEVEL DESIGN OF POSSIBLE SOLUTIONS.....	11
<i>Option 1: Link Current Spreadsheets to Solution</i>	<i>11</i>
<i>Option 2: Copy Data from Spreadsheets into Solution.....</i>	<i>12</i>
RECOMMENDATION FOR HOW TO GATHER DATA FROM SPREADSHEETS	13
ACRONYMS	13

Problem Statement: Obtaining Data for Cash Flow Forecasting

Omnitech's accounting system, QuickBooks (QB), does not provide the functionality for cash flow forecasting. To forecast cash flows, employees must manually update several spreadsheets to handle the payment terms for purchases and other data needed to predict cash flows. This system is confusing, time consuming, and duplicates efforts for parties in charge of keeping the system updated and accurate.

Omnitech needs a system that updates cash flows automatically, eliminates excess steps in cash flow calculations, organizes data, and presents reports every week or upon demand.

During our requirements and analysis phase of this project, our team has discovered several problems needing solutions. During the analysis of these problems, we found several solutions to each. Per client request, we have analyzed each of the possible solutions and listed pros and cons for each. We will present our findings to the client so they can choose the best options for them. If the choice on what solution to take is clear, we will proceed to design the solution further and begin implementation. If there is no clear choice, we will provide more in-depth analysis and do "prototype implementation" of each solution.

Problem: Handling Complex Terms

Payment terms are the manner and schedule in which a company pays cash for purchases. Payment terms are critical for calculating cash flows; hence, they need to be included in our system. QB has a field for users to specify terms of a purchase, but QB can only make calculations of payment dates for simple terms such as "Net 30" (full payment 30 days after the delivery of goods). However, Omnitech uses some more complex payment terms for which QB cannot calculate payment dates. Therefore, Omnitech calculates the cash flows by manually entering terms data, shipping dates, transaction dates, etc. in a spreadsheet. The manual entry of terms data often means that forecasts are outdated and incorrect.

To automate the cash flow calculations, our solution must manage the complex terms so that users need only enter terms data once and allow users to make changes to terms arrangements when desired. An outside program must do the calculations of cash flows using the terms since QB cannot make the appropriate calculations. We have discovered three solutions to use for entering and updating complex terms for later calculations.

Analysis and High Level Design of Possible Solutions

Option 1: Add Complex Terms to Current Terms Field in QB

QB has a field for terms in the purchase order form with predefined, commonly used terms such as "Net 30". QB has calculations for payment dates for these predefined terms. Users may add new terms to QB, but QB does not accept enough information from the user to perform calculations for the more complex terms. This means that the only useful information the user can enter is the terms title (see Figure 9).

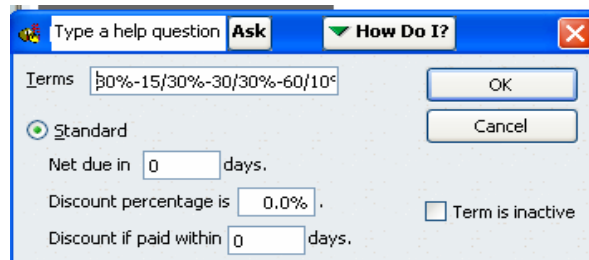


Figure 9: This figure shows the window that opens when users want to add a new terms title. Unfortunately, users cannot input enough information related to complex terms for QB to make the right calculations.

After the users add a complex terms title to QB, they may select that title from a drop-down box any time in the future (see Figure 10). The system set up to automate cash flow calculations will extract official, up-to-date data (including terms title) needed for cash flow calculations. The system will use different functions for different terms titles to calculate cash flows. When users create new terms, they must also enter information into the system so the system can generate the appropriate functions. The solution will create the functions based in the information the users input (shipping dates, etc.). Users can change terms for purchase orders in QB and change shipping dates in the system.

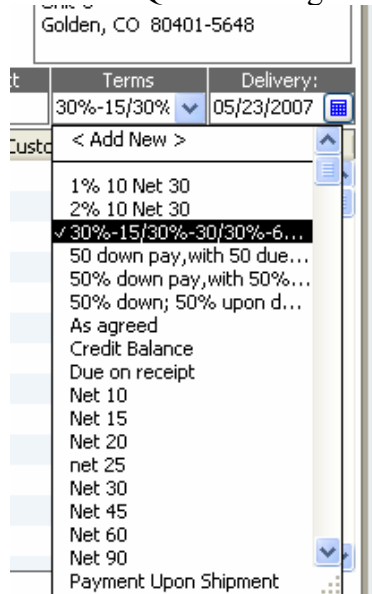


Figure 10: This figure displays the current terms field in the QB purchase orders form. The field has a drop-down box and the user has selected the newly added complex terms.

Pros

- Easiest for users
 - Users continue to use QB
 - No duplication of data entry
- Drop-down box reduces risk of operator errors in entering terms

Cons

- Users must remember to add new terms into system after adding to QB

- Changes to dates must be made in system

Reflection of Pros and Cons

This system is the easiest for the user since they only have to enter new terms titles into QB once and can select the already entered terms titles in a drop-down menu. They will continue to use QB for the most part, with which they are familiar.

The users have to go to a separate system to enter information regarding terms newly added into QB. The users may either forget to do this step and they will have to take time to become familiar with the system. The user must also make changes to the terms dates of payments in the system. This solution only reduces the amount of data entry in a separate location it does not eliminate it.

Option 2: Create New Terms Field in QB

QB gives users the option to add new fields (and name them) and remove existing fields. We will remove the current terms field and create a new terms field where users can enter terms data manually for each purchase order (see Figure 11). If the users enter new terms data (not already in the system) in the field, they must enter extra information into the system so the system will generate functions for that set of terms. Users only need to enter the extra information once into the system and the system will use the same function for the same terms in the future.

The screenshot shows a purchase order form with the following fields and values:

- Template: PO1
- Date: 05/23/2007
- P.O. No.: OCS21459
- Ship To: Omnitech International, Inc., 500 Corporate Circle, Unit O, Golden, CO 80401-5648
- OII Project: [Empty]
- Delivery: 05/23/2007
- Terms: 30%-15/30%-...
- Customer: [Empty]
- Amount: [Empty]

Figure 11: This figure shows the new terms field (without drop-box) where users enter terms data manually for each purchase order.

Pros

- User does not have to go to a new QB window to enter new terms

Cons

- User enters data manually
 - Risk of mistake
- Most difficult for user
 - Re-enter data for terms

- Users have to check to see if terms are new terms or terms already in system

Reflection of Pros and Cons

The user does not have to go to a new window in QB to enter new terms, but the user has to enter the terms data by hand every time. Manual data entry increases the risk of having typos and it takes more time than selecting terms from a drop-down menu. Additionally, the user must check the system or a list somewhere else to see if the terms are already in the system or if they are new. If the terms are new, the user must enter additional information into the system for proper calculations to take place.

Option 3: Users Enter Data into Solution

The user will enter all normal purchase order data in QB, except terms data, which users will enter into the system. The RefNumber (each purchase order has a RefNumber) will allow us to associate terms entered into the system with the other purchase order data for that purchase entered into QB. The users will enter the terms data into a form or table and once the users enter a term into the system once, they can select the term from a drop-down box in the future. The terms data entered in the system will include percentages of the total payment and payment dates (see Figure 12). The system will have all the terms data needed to generate functions based on what the user enters. The system still must extract other purchase order data from QB to finish the calculations.

TermPORelations : Table								
RefNumber	Term1	Term1Date	Term2	Term2Date	Term3	Term3Date	Term4	Term4Date
▶ 19	25	5/10/1955	50	5/15/1955	25	5/30/2005		
200	50	6/12/1995	50	7/10/1995				
22	100	9/9/1999						
7	30	4/13/1985	30	4/30/1985	40	5/15/1985		
*								

Figure 12: This is an example of a table the user would enter terms data to including different terms percentages, different payment dates, and the RefNumber so that the terms are associated with the proper purchase.

Pros

- User friendly
 - Terms data is all entered into one place

Cons

- Medium difficulty for user
 - User must enter more data outside of QB

Reflection of Pros and Cons

The user interface of the system will accommodate the users and they will adjust quickly. The users will enter any terms data necessary into just one location instead of maybe two.

The user will have to learn to use the new system and spend more time between two data entry locations.

Option 3A: Users Enter All Cash Flow Data into Solution

In Option 2 (link data between QB and solution) of Gathering Data from QuickBooks, the user will enter all purchase order data (including terms) into the system. The system will connect to and update the purchase order data in QB when users enter data into the system. If Omnitech chooses Option 2 of Gathering Data from QuickBooks, then Option 3 is the clear choice for handling complex terms data.

Pros

- User enters all cash flow data in one location

Cons

- Bigger learning curve for users
- Risk of QB not receiving the data it needs
 - If this happens, user must re-enter data into QB

Recommendation for Handling Complex Terms

- We do not recommend Option 2 (Create New Terms Field in QB) because it is similar to Option 1 (Add Complex Terms to Current Terms Field in QB), but is more difficult for the user
- We do not recommend Option 3 (Users Enter Data into Solution) because it is more work for the users
- We recommend Option 1 (Add Complex Terms to Current Terms Field in QB)
 - Easiest for user
 - User still works with QuickBooks
- We recommend Option 3A if Option 2 for Gathering Data from QuickBooks is chosen

Problem: Gathering Data from QuickBooks

The majority of the data that Omnitech needs to predict cash flows is already in QuickBooks; however, getting the data from QuickBooks is a nightmare for Omnitech's employees. Currently, users create reports in QuickBooks and export them into Excel. The users then manipulate the data so that it works with the cash flow forecasting system; sometime this requires manual data entry. Due to the enormous amount of data that has to be processed, this method takes forever. Omnitech needs a better, automatic way of getting the data from QuickBooks in order for cash flow calculations to occur.

Analysis and High Level Design of Possible Solutions

Option 1: Link Data from QB to Solution

In this solution, the employees will enter the majority of the data (PO data w/ terms, cash-in data, and cash-out data) into QuickBooks; however, if a complex term does not exist, users will need to define it in our solution and possibly enter the terms title in QuickBooks. The QuickBooks Open Database Connectivity (QODBC) will then link the data from QB to the data in the solution (see Figure 13).

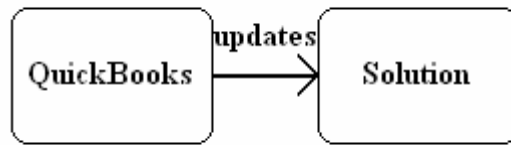


Figure 13: The QODBC driver that OII currently has will allow QB to link data to the solution.

Pros

- Easiest for user
 - Users continue to use QB
 - Less switching between programs

Cons

- No way to ensure that users enter terms data into solution
 - Cash flow forecast will be wrong

Reflection of Pros

It is not the purpose of our solution to replace QuickBooks. To be more precise, our solution will supplement QuickBooks by handling the things that it cannot. Omnitech’s employees are already familiar with QuickBooks, and use it for many different tasks other than cash flows. For this reason, it will be much easier for employees to continue to use QuickBooks as much as possible, so that they do not have to switch between two programs.

Reflection of Cons

With this solution, there is no way to be certain that users will enter terms data into our solution as well as QuickBooks. This will cause an error in the calculations, but an error message will tell users that they must enter terms data.

Option 1A: Link Data from QB to MS Access

QuickBooks will link data in real-time to Access using QODBC. We will create forms for users to input and change terms data. We will create several reports formatted to fit the users’ needs (canned reports). Access will generate the preformatted reports on demand with up-to-date data. Users can generate the reports within Access by clicking a button or VB Script (a scripting language for MS products) will generate reports on a designated time interval. Access also includes a Wizard that allows users to create their own reports and a design view to modify existing reports.

Pros

- Easy for users
 - Input terms data in user-friendly forms
 - Reports are easy for users to use in Access

Cons

- Learning curve
- Users may want to work with data themselves
 - Users will have to export data to Excel

Reflection of Pros

Microsoft Access serves as a great front-end for programs. Forms in Access are intuitive and are easy for employees to use. Buttons and drop down boxes will do most of the work for them. Reports are very easy for users to create in Access. The user may choose from several canned reports, or he or she may create custom reports using the Report Wizard.

Reflection of Cons

Users will have to overcome a small learning curve with this system. Most of Omnitech's employees have little or no experience with MS Access. Users would need to learn some of the basics of Access before they would feel comfortable with the system.

Some of Omnitech's employees may want to work with the data themselves. It is most likely that they will use Excel rather than Access, so they will need to export the data that they need to Excel. This may lead back to the problems that Omnitech is having with QuickBooks, so it is important that our system will export data in a way that will be useful for employees in the future.

Option 1B: Link Data from QB to MS Excel

QuickBooks will link data to Excel using QODBC. Users will need to refresh the data by clicking a button in Excel or VB Script can update data on designated intervals. We will create forms for users to input and change terms data. We will create several reports formatted to fit the users' needs.

Pros

- Users are familiar with Excel
 - Users can manipulate formulas / results / report formats by themselves

Cons

- Linked data is not real-time
- Inability to create users
 - Little control over user access
- Only one person can modify spreadsheet at a time

Reflection of Pros

Users are already comfortable with Excel and know how to work and control it.

Reflection of Cons

There is no way in Excel to create a true "real-time" solution for linking data with QuickBooks like there is in the other solutions. Users will have to click a button to refresh data, or there will have to be a timer that tells the program to refresh the data (using VB Script).

In Excel, there is no way to create users and groups. Therefore, it is impossible to allow some users to do certain things, but block others. This means that users who are experienced with ODBC connections may be able to view employee's private data, including social security numbers, salary information, and more.

Option 1C: Link Data from QB to MySQL Database

QuickBooks will link data in real-time to the MySQL database using QODBC. We will create forms for users to input and change terms data from the web. We will create several reports formatted to fit the users' needs. PHP will generate the preformatted reports on demand with up-to-date data from the MySQL database. These reports will be available on the web and users can save, print, email, fax, and export reports. PHP can generate these reports on a designated time interval.

Pros

- Easy for user
- Works well with Item Master Database
- No need for VPN when working remotely

Cons

- Maintenance requires programmer

Reflection of Pros

Almost all employees today are proficient with web applications, so it would be relatively easy for them to use a web interface for their cash flow forecasting system. Users would use forms to enter the data that the system needs. Canned reports will allow employees to create common reports effortlessly. Users can use a Wizard to create additional reports.

Since this solution is web-based, it will work well with Omnitech's Item Master Database¹⁵, which uses a PHP web interface and a MySQL database. The solution will also be accessible worldwide, allowing users in Malaysia, Poland, and other parts of the world to use it as well without using a VPN.

Reflection of Cons

A major disadvantage to this solution is that it requires a programmer to maintain the system. If Omnitech needs new reports, the Wizard may not be able to create them. In such cases, users will not be able to create reports themselves and will have to wait until a programmer can create the reports needed. Users may work around this by exporting data to Excel.

Option 2: Link Data between QB and Solution

In this solution, the users will enter all cash flow related data into our system. Our system will then update QuickBooks with the data that it needs using an additional

¹⁵ The Item Master Database keeps track of all of Omnitech's purchased and manufactured parts, documents, vendors, manufacturers, etc. and it will be used to create Bills of Materials (records of parts required for assemblies), track shipping, track prices, and more.

QODBC driver that will cost \$500.00 (see Figure 14). QuickBooks will still send updates done in QuickBooks to the solution.

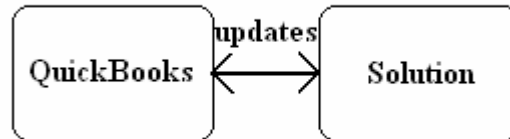


Figure 14: After purchasing the extra QODBC driver, QB and the solution can link data to each other.

Pros

- Users enter all cash flow data into one location

Cons

- Bigger learning curve for users
- Less convenient and not as easy for users
- Risk of QB not receiving the data it needs
 - If this happens, user must re-enter data into QB
- \$500 driver
 - However, this driver will probably be needed for the Item Master Database

Reflection of Pros

In this solution, users will enter cash flow related data into one location. This will be easier for users in certain situations. For example, users will only have to create new terms in the system; the system will update QuickBooks automatically.

Reflection of Cons

There is a considerably bigger learning curve for the users with this solution. They will not only have to learn how to enter terms into the new system, they will have to learn how to enter all cash flow related data into the system as well. Although we can make the forms look very similar to QuickBooks, it will not be the same.

Since users will still be entering other data into QuickBooks, they must switch back and forth between the two systems.

In this solution, there is also a chance that the system will not enter data correctly into QuickBooks in every situation. It may not be apparent initially that some transactions require our system to update additional tables in QuickBooks. This type of error is both difficult to predict, and difficult to diagnose since the error may show up weeks later.

Option 2A: Link Data between QB and MS Access

Access will link the data that users input into forms back to QuickBooks in real-time using QODBC. Users enter data and generate reports in the same fashion as Option 1A.

Pros and Cons

Same as Option 1A

Option 2B: Link Data between QB and MS Excel

Excel will link the data that users input into forms back to QuickBooks using QODBC. The users will need to perform additional steps to update the QuickBooks data. Users enter data and generate reports in the same fashion as Option 1B.

Pros and Cons

Same as Option 1B

Option 2C: Link Data between QB and MySQL Database

PHP will link the data that users input into forms back to QuickBooks in real-time using QODBC. Users enter data and generate reports in the same fashion as Option 1C.

Pros and Cons

Same as Option 1C

Recommendation for Gathering Data from QuickBooks

- We do not recommend Option 2 (Linking Data between QB and the Solution)
 - It is more work for users because more data is entered into two different locations
 - It is risky because it difficult to know if all data is being entered into QuickBooks correctly by our system
- We recommend Option 1A or Option 1C
 - Option 1A (Linking Data from QB to MS Access)
 - Users can “do it themselves”
 - They can create reports, edit formulas, change formats, perform maintenance, etc.
 - Option 1C (Linking Data from QB to MySQL Database)
 - This solution provides a web interface, which allows users to work remotely without using a VPN

Problem: Gathering Data from Spreadsheets

Some of the data that Omnitech needs for cash flow forecasting comes from several different spreadsheets that employees maintain throughout the office. Users copy mutual data from one worksheet to another and from QuickBooks then format it to match their spreadsheets. This is very inefficient to upkeep. In order to simplify the process, our solution needs to establish a better way for users to enter and gather data.

Analysis and High Level Design of Possible Solutions

Option 1: Link Current Spreadsheets to Solution

The system will collect “official” data (accurate, up-to-date) from the appropriate spreadsheets and QuickBooks and perform cash flow calculations afterwards. The users will continue to use their spreadsheets, so the format must remain the same. In order to improve efficiency, we must automate data collection between the spreadsheets and QuickBooks (see Figure 15).

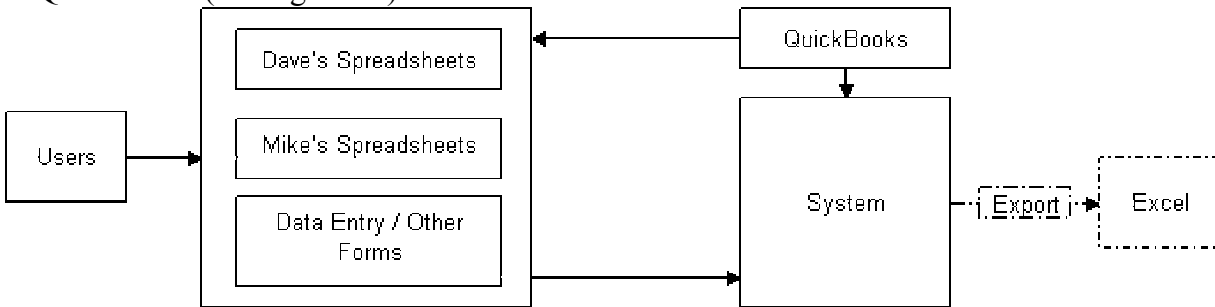


Figure 15: Users will enter data in their current spreadsheets. Their spreadsheets will receive data from QuickBooks and other spreadsheets “automatically”. Our system will collect data from the users’ spreadsheets and QuickBooks and perform calculations. The users have the option to export data from the system to Excel.

Pros

- Users are comfortable with their current spreadsheets

Cons

- Solution possibly unreliable
 - Links can break
- Data is scattered

Reflection of Cons

Keeping the data in spreadsheets may be an unreliable solution to this problem for several reasons. To begin, links can be broken very easily (even with protection in place). A link can be broken if a user deletes rows or columns. Links will also be broken if files are renamed or moved. In addition, Excel only has limited error-checking capabilities for data entry. If a user does not enter something correctly, the programmer may not have a way to catch it before it goes into the system.

There are also several risks involved when the data is scattered throughout the office. One risk is that if somebody leaves the company, does anybody else know where their data is and how to maintain it? It may be difficult to figure out if nobody knows. Another is that it is very hard to add functionality in the future. The programmer, like us, would need to learn where all of the data comes from and why it is important. Omnitech should not take this risk lightly since they would also like to use this system to track shipping and receiving in the future.

Option 2: Copy Data from Spreadsheets into Solution

In this solution, we will set up the back-end (calculations, data storage, inner workings, etc.). We will then create the forms and reports that users need to enter the same data as in their spreadsheets. When the back-end is set up and forms created, we will import the necessary data from the users' spreadsheets into our system. From then on, users will enter their data into our system by using the forms. Data will still link from QuickBooks to the solution. The users will then be able to create reports with our system and export them to Excel if they want to do further analysis on them (see Figure 16).

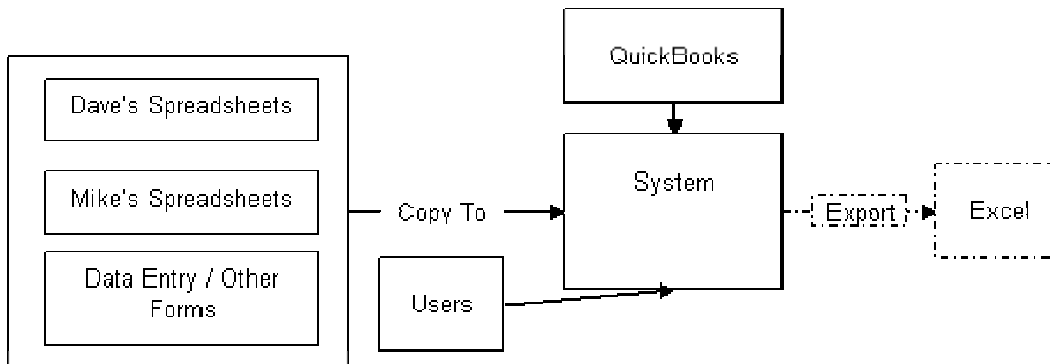


Figure 16: We will import the data from spreadsheets into our system. From then on, users will enter data into our system by using forms. The users will then be able to create reports with our system and export to Excel if they want to do further analysis.

Pros

- Data is in one place
 - Long-term solution

Cons

- Time to implement

Reflection of Pros

Since all data storage, data entry, and system functionality is in one place, this is more likely to be a long-term solution for Omnitech. To begin, users will know that everything related to cash flows and budgeting is in the system, saving them the time of asking around to see where everything is. This is especially helpful for users who are new to the system. In turn, if the user cannot find the report that they need in the system, it will be much easier for someone to create additional reports.

Reflection of Cons

One disadvantage to this solution is the time that it will take to implement. First, the programmer will need create the back-end for this solution. This could be many linked Excel spreadsheets or some kind of database solution like MS Access, MySQL, or MSSQL.

Recommendation for How to Gather Data from Spreadsheets

- We do not recommend Option 1 (Linking Current Spreadsheets to the Solution)
 - It is not a long-term solution
 - It is not as reliable as Option 2 (Copying Data from Spreadsheets into the Solution)
 - It will take new users a long time to become familiar with this system
- We recommend Option 2 (Copying Data from Spreadsheets into the Solution)
 - Everything is in one location, making it easier to manage and add new functionality later
 - This makes it a long-term solution, which is important since Omnitech would like to use this system for other tasks as well

Acronyms

DB – Database

OII – Omnitech International, Inc.

PO – Purchase orders

QODBC – QuickBooks Open Database Connectivity

QB - QuickBooks

MS - Microsoft

Appendix B: Improving Budget Management and Cash Flow Forecasting for Projects Presentation



Improving Budget Management and Cash Flow Forecasting for Projects

Kurtis Griess
Nick Henry

June 05, 2007

1

Omnitech's Cash Forecasting and Budget Management Systems Need an Upgrade

- Problems with current system
 - Capturing actuals is extremely difficult and time consuming
 - QuickBooks cannot handle complex terms of payment
 - Inconsistent tracking categories used by different users/ systems
 - Multiple spreadsheets
 - Repeated data entry
- Features of solution
 - Email, print, and fax reports
 - Automatic updates with actual cost data
 - Screen-viewed reports
 - Ability to extract for further user analysis
 - Achieve "single source of the truth"

2

The Solution must Handle Complex Terms

- Why
 - Critical for calculating cash flow
 - Payment dates are currently manually entered in spreadsheets
 - Forecasts are often outdated and incorrect
 - No automated tie to actual cash-out or payment trigger date changes
- How
 - Several options
 - Choose best for OII

3

There are Three Options to Handle Complex Terms

- Option 1: Add Complex Terms to Current Terms Field in QB
- Option 2: Create New Terms Field in QB
- Option 3: Users Enter Terms Data into Solution for Every PO
- Recommendation

4

There are Three Options to Handle Complex Terms

- **Option 1: Add Complex Terms to Current Terms Field in QB**
- Option 2: Create New Terms Field in QB
- Option 3: Users Enter Terms Data into Solution for Every PO
- Recommendation

5

Option 1: Add Complex Terms to Current Terms Field in QB

Description:

- Users can add new terms titles in QB (see picture below)
- After the title is added to QB once, it can be selected in the future by the user from a drop-down menu

6

Option 1: Add Complex Terms to Current Terms Field in QB

Description continued:

- The solution will perform calculations for cash flows based on the terms titles entered by the user in QB
- Users will add extra data into the solution when new terms are added to QB
- The calculations will take place in the solution (outside of QB)

Pros

- Easiest for user
 - Drop-down menu
- Users continue to use QB for the majority of the time

Cons

- Users might forget to add new terms into solution after adding to QB
- Users must switch between programs

7

There are Three Options to Handle Complex Terms

- Option 1: Add Complex Terms to Current Terms Field in QB
- **Option 2: Create New Terms Field in QB**
- Option 3: Users Enter Terms Data into Solution for Every PO
- Recommendation

8

Option 2: Create New Terms Field in QB

Description:

- Create a new terms field for the purchase orders window in QB and get rid of the current terms field
- Users will enter terms manually into the new terms field in QB for every purchase order
- Users will add extra data into the solution when new terms are added to QB

9

Option 2: Create New Terms Field in QB

Description recap:

- Create a new terms field for the purchase orders window in QB and get rid of the current terms field
- Users will enter terms manually into the new terms field in QB for every purchase order
- Users will add "extra data" into the solution when new terms are added to QB

Pros

- Users do not have to go to a different QB screen to add new terms

Cons

- Users enter data manually
- Most difficult for user
 - Users must check solution often to see if "extra data" has been input
- Most risky

10

There are Three Options to Handle Complex Terms

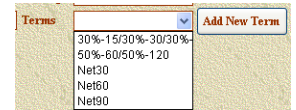
- Option 1: Add Complex Terms to Current Terms Field in QB
- Option 2: Create New Terms Field in QB
- **Option 3: Users Enter Terms Data into Solution for Every PO**
- Recommendation

11

Option 3: Users Enter Terms Data into Solution for Every PO

Description:

- The user will enter terms data (title, other info for un-encountered terms) into the solution and enter other PO data in QB for every PO
- After terms are established, users will select terms data from drop-down boxes



TermsTitle	NumDays1	Percentage1	NumDays2	Percentage2	NumDays3	Percentage3	NumDays4	Percentage4
Net30	30	100	0	0	0	0	0	0
Net60	60	100	0	0	0	0	0	0
Net90	90	100	0	0	0	0	0	0
30%-15/30%-30/30%-60/10%-90	15	30	30	30	60	30	90	10
50%-60/50%-120	60	50	120	50	0	0	0	0
	0	0	0	0	0	0	0	0

12

Option 3: Users Enter Terms Data into Solution for Every PO

Description recap:

- The user will enter terms data (title, other info for un-encountered terms) into the solution and enter other PO data in QB for every PO
- After terms are established, users will select terms data from drop-down boxes

Pros

- User friendly
 - Drop-down boxes will be used most of the time

Cons

- Medium difficulty for user
 - Must work between QB and solution more often

13

There are Three Options to Handle Complex Terms

- Option 1: Add Complex Terms to Current Terms Field in QB
- Option 2: Create New Terms Field in QB
- Option 3: Users Enter All Terms Data into Solution for Every PO
- **Recommendation**

14

Recommendation for Handling Complex Terms

- We do not recommend Option 2 (New Terms Field) because it is similar to Option 1 (Current Terms Field), but is more difficult for the user
- We do not recommend Option 3 (Data Entered to Solution) because it is more unnecessary work and learning for the users
- We recommend Option 1
 - Easiest for user
 - User still works with QuickBooks

15

To Automate the Forecast and Budget Status, the Solution must Gather and Connect the Data

- Where
 - QuickBooks
 - Spreadsheets
 - Possibly others
- Why
 - Data is needed for the calculations and budget review
- How
 - Several options
 - Choose best solution for OII

16

To Automate the Forecast and Budget Status, the Solution must Gather and Connect the Data

- Where
 - QuickBooks
 - Spreadsheets
 - Possibly others
- Why
 - Data is needed for the calculations and budget review
- How
 - Several options
 - Choose best solution for OII

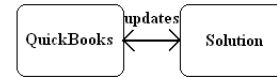
17

There are Two Options to Gather Data from QuickBooks

- Option 1: Link data from QB to solution



- Option 2: Link data between solution and QB

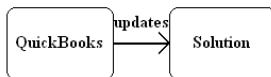


- Recommendation

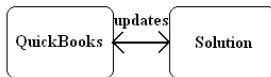
18

There are Two Options to Gather Data from QuickBooks

- Option 1: Link data from QB to solution



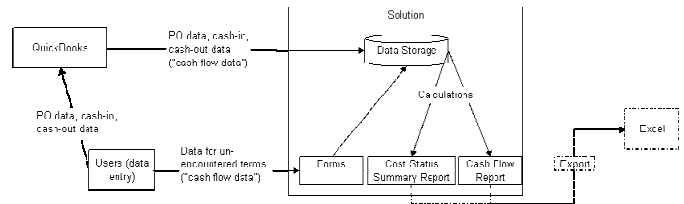
- Option 2: Link data between solution and QB



- Recommendation

19

Option 1: Link Data from QB to Solution



Users (data entry):

- Jodi, Mike, and Roy

Reports users:

- Karen and Dave (project manager)
- Possibly Jodi, Mike, and Roy

20

Option 1: Link Data from QB to Solution

Description:

- Users enter PO data with terms, cash-out (checks, EFT's, etc.), and cash-in (revenue received) into QB and some data for terms into the solution
- Data is linked from QB to solution (using QODBC)
- Link works differently for different solutions (sub-options)
 - Option A: MS Access
 - Option B: MS Excel
 - Option C: MySQL Database

Pros

- Easiest for user
 - User continues to use QB most of the time

Cons

- No feedback mechanism to verify that data was received correctly
- Not possible to guide users from QB to solution to add new terms

21

Option 1A: Link Data from QB to MS Access

Description:

- Data is updated in real-time when changes are made in QB
- Users enter some data for terms into a form from inside Access
 - Data only required when previously unknown terms are encountered
- Reports are prepared and viewed in Access and can be exported to Excel

Pros

- Easy for user
 - Access has built-in report generation/saving capabilities
 - Users view data different ways
- No programmer needed

Cons

- Small learning curve
- Users may want to work with data themselves
 - Users will have to export data to Excel

22

Option 1B: Link Data from QB to MS Excel

Description:

- Users must refresh data in spreadsheet in order to capture updates made in QB
- Users enter some data for terms into a spreadsheet
- Reports are prepared and viewed in Excel

Pros

- Users are familiar with Excel
- Users can manipulate formulas, results, and report formats

Cons

- Users must refresh data manually
- Less control over user access
- Only moderately easy for user
- Single user access
- Manipulating results can create new "sources of truth" ²³

Option 1C: Link Data from QB to MySQL Database

Description:

- Data is updated in real-time when changes are made in QB
- Users must enter some terms data into a web form
- Reports are prepared by a programmer
- Users can generate and view reports on the web and can export to Excel

Pros

- Easy for user
- Works well with Item Master Database
- No need for VPN when working remotely

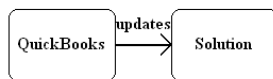
Cons

- Maintenance and report building requires programmer
- Users may want to work with data themselves
 - Users will have to export data to Excel

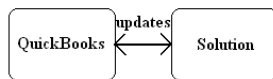
24

There are Two Options to Gather Data from QuickBooks

- Option 1: Link data from QB to solution



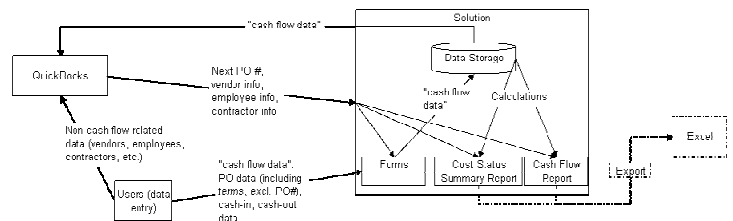
- Option 2: Link data between solution and QB



- Recommendation

25

Option 2: Link Data between Solution and QB



Users (data entry):

- Jodi, Mike, and Roy

Reports users:

- Karen and Dave (project manager)
- Possibly Jodi, Mike, and Roy ²⁶

Option 2: Link Data between Solution and QB

Description:

- Users enter PO data (terms data, selected vendor, etc.), cash-out (checks, EFT's, etc.), and cash-in (revenue received) into the solution in forms
- Solution updates user entered data to QB through link
- Solution receives new PO numbers, vendor's information, and other information from QB through link
- To link in both directions, a \$500 driver is needed
 - \$500 driver is needed for automating the "shopping cart" from the Item Master into a QB PO

27

Option 2: Link Data between QB and Solution

Description:

- Link works differently for different solutions (sub-options)
 - Option A: MS Access
 - Option B: MS Excel
 - Option C: MySQL Database
- PO changes, cash-in, and cash-out changes are made in solution

Pros

- Users enter all cash flow data into one location
- Has potential to work with Item Master Database

Cons

- Bigger learning curve for users
- Less convenient and not as easy for users
- Risk of QB not receiving the data it needs
 - If this happens, user must re-enter data into QB ²⁸

28

Option 2A: Link Data between QB and MS Access

Description:

- Data is updated in real-time
 - QB data automatically links to Access
 - Access data automatically links to QB
- Users enter all “cash flow related” data into a form from inside Access

Pros

- Data entry form is easy for user
 - Drop-downs for data that is in the system
 - Checkboxes, radio-buttons

Cons

- Users must learn some Access
- Users may want to work with data themselves
 - Users will have to export data to Excel

29

Option 2B: Link Data between QB and MS Excel

Description:

- Data entered in QB or Excel requires users to manually refresh to capture updates from one program or the other
 - User clicks refresh button in Excel
- Users enter all “cash flow related” data into a spreadsheet

Pros

- Users are familiar with Excel
- Users can manipulate formulas, results, and report formats

Cons

- Users must refresh data manually
- Lack of permissions
 - Little control over user access
- Only moderately easy for user
- Single user access

30

Option 2C: Link Data between QB and MySQL Database

Description:

- Data is updated in real-time
 - QB data automatically links to MySQL database
 - MySQL database automatically links to QB
- Users must enter all “cash flow related” data into a web form

Pros

- Web form is easy for users
- Works well with Item Master Database
- No need for VPN when working remotely

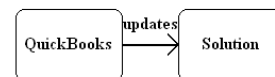
Cons

- Maintenance requires programmer
- Users may want to work with data themselves
 - Users will have to export data to Excel

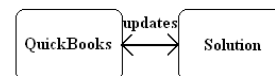
31

There are Two Options to Gather Data from QuickBooks

- Option 1: Link data from QB to solution



- Option 2: Link data between solution and QB



- **Recommendation**

32

Recommendation for Gathering Data from QuickBooks

- We do not recommend linking data between QB and the solution (Option 2)
 - More work for users of QuickBooks
 - Users will have to switch back and forth between the two
 - Riskier...
 - Difficult to know if all data is being entered into QuickBooks correctly by our system
- We recommend Option 1A or Option 1C
 - Option 1A – Link data from QB to Access
 - Users can “do it themselves”
 - Reports, edit formulas, maintenance, etc.
 - Option 1C – Link data from QB to MySQL Database
 - Web interface
 - Users do not need QB

33

To Automate the Forecast, the Solution must Gather Data

- Where
 - QuickBooks
 - **Spreadsheets**
 - Possibly others
- Why
 - Data is needed for the calculations
- How
 - Several options
 - Choose best solution for OII

34

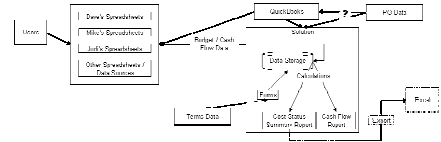
The Solution must Incorporate Data from Current Spreadsheets

- Why
 - Data is needed for cash flow calculations
 - Improve cash flow forecasting and budget management processes
 - Reduce time
 - Reduce efforts
 - Reduce risk
- Where
 - Dave's spreadsheets
 - Budget data
 - Mike's spreadsheets
 - Contains "well-maintained" ship date
 - Other sources
 - Jodi's spreadsheet
 - BigTime

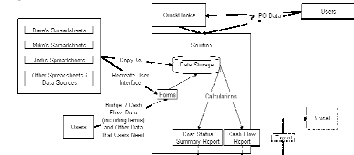
35

There are Two Ways to Gather Data from Spreadsheets

Option 1: Link Current Spreadsheets to Solution



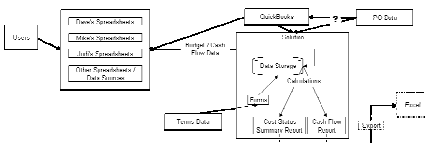
Option 2: Copy Data from Spreadsheets Into Solution



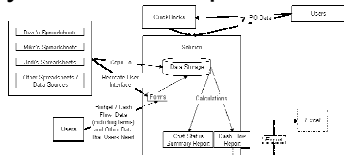
36

There are Two Ways to Gather Data from Spreadsheets

Option 1: Link Current Spreadsheets to Solution



Option 2: Copy Data from Spreadsheets Into Solution



37

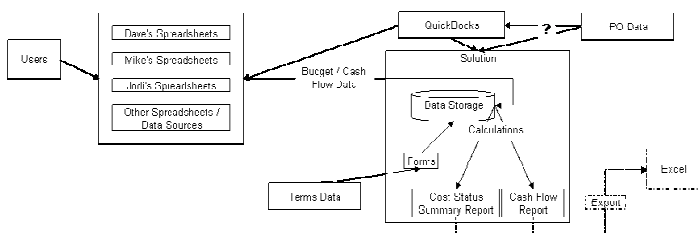
Option 1: Link Current Spreadsheets to Solution

Description:

- Users continue to enter data into their own spreadsheets
 - Spreadsheets modified to improve efficiency (drop downs, data from QB, data from other spreadsheets, data from solution, etc.)
- QuickBooks links data to user spreadsheets to minimize retyping data
- Solution will gather data by linking to these spreadsheets

38

Option 1: Link Current Spreadsheets to Solution



Users (data entry):

- Jodi, Mike, and Roy

Reports users:

- Karen and Dave (project manager)
- Possibly Jodi, Mike, and Roy

39

Option 1: Link Current Spreadsheets to Solution

Description recap:

- Users continue to enter data into their own spreadsheets
 - Spreadsheets modified to improve efficiency (drop downs, data from QB, data from other spreadsheets, data from solution, etc.)
- QuickBooks links data to user spreadsheets to minimize retyping data
- Solution will gather data by linking to these spreadsheets

Pros

- Users are comfortable with their current spreadsheets
- Users can manipulate formulas, results, and report formats

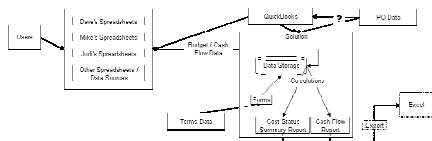
Cons

- Solution possibly unreliable
 - Links can break
- Not as easy to use
- Data is scattered

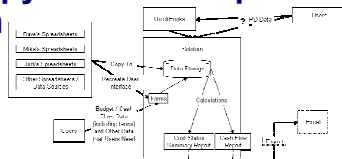
40

There are Two Ways to Gather Data from Spreadsheets

Option 1: Link Current Spreadsheets to Solution



Option 2: Copy Data from Spreadsheets Into Solution



41

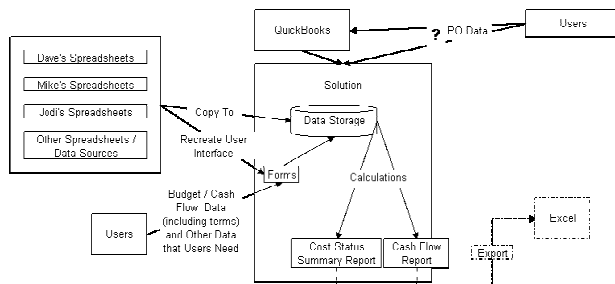
Option 2: Copy Data from Spreadsheets Into Solution

Description:

- Forms and reports are created, formulas working, etc.
- Necessary data will be copied (imported) into the solution
- From then on, users will use the solution to:
 - Enter / modify their data
 - Create Reports (solution will do most of the work)

42

Option 2: Copy Data from Spreadsheets Into Solution



Users (data entry):

- Jodi, Mike, and Roy

Reports users:

- Karen and Dave (project manager)
- Possibly Jodi, Mike, and Roy

43

Option 2: Copy Data from Spreadsheets Into Solution

Description:

- Forms and reports are created, formulas working, etc.
- Necessary data will be copied (imported) into the solution
- From then on, users will use the solution to:
 - Enter / modify their data
 - Create Reports (solution will do most of the work)

Pros

- Easier for users to update data
- Data is in one place
- Long-term solution

Cons

- Time to implement
- Users must adjust to the new system

44

Recommendation for How to Gather Data from Spreadsheets

- We do not recommend "Linking Current Spreadsheets to Our Solution" (Option 1)
 - Not as reliable
 - Data is scattered
- We recommend "Copying Data From Current Spreadsheets Into Our Solution" (Option 2)
 - Data entered and managed from one location
 - Easier for users
 - Long-term solution

45

This Presentation of Solutions took place on June 5 and the Outcomes are Highlighted in the Following Slides

Attendees included

- Karen Buelow, CFO
- Dave Harrison, V.P./General Manager
- Jim Weed, Senior Manufacturing Engineer
- Shelley Walton, Controller

46

The Outcome for Handling Complex Terms Follows

- Option 1: Add Complex Terms to Current Terms Field in QB
 - This may work in the short run
- ~~Option 2: Create New Terms Field in QB~~
- Option 3: Users Enter Terms Data into Solution for Every PO*
 - We may use this solution with a slight variation (see next slide)

*Preferred Option pending further analysis.

47

Handling Terms Option 3 Variation: Enter PO in QB, Enter New Terms in Solution

- Users enter all usual PO data in QB
- Users check if terms title is in the drop-down menu
 - If the terms title is available, user will select the terms and continue as usual
 - If the terms title is not available in the drop-down menu
 - Users go to the solution to enter terms data (title, trigger dates, etc.)
 - Solution links to QB and sends terms titles back to QB for that PO and future selection from the drop-down menu

48

The Outcome for Gathering Data from QuickBooks Follows

- Option 1: Link data from QB to solution
 - Short-term solution: quick and easy to implement, but too restrictive
- Option 2: Link data between solution and QB*
 - Explore further (e.g. enter new terms data only into the solution and then link updates to QB)
 - Long-term solution: data entry is in one place, easier to add more functionality, will work with Item Master DB

*Preferred Option pending further analysis.

49

The Outcome for Gathering Data from Spreadsheets Follows

- ~~Option 1: Link Current Spreadsheets to Solution~~
 - Too “fragile” and easily broken
- Option 2: Copy Data from Spreadsheets Into Solution
 - Helps create “single source of truth”
 - Easy for current and future users

50

The Solution will use Either Access or MySQL

- ~~MS Excel~~
- MS Access
 - Needs VPN (connection is more difficult)
 - See if Access can create reports for MySQL
- MySQL*
 - Look into reporting capabilities with Crystal
 - Security is an issue
 - On the web
 - Final users and data entry users need access

*Preferred Option; however, needs more flexibility in user generated reports as provided by Access

51

Appendix C: Requested CSSR Template

DESCRIPTION	VENDOR SELECTED	P.O. ISSUE ORDER DATE	PAYMENT TERMS	ANTICIPATED SHIP DATE	CONTRACT PROJECT VALUE (PRORATA)	20% INVOICE UPON SHIPMENT (REVENUE)	PROJECT BUDGET (ESTIMATED COST)	PURCHASE COMMITMENT (ACTUALS)	CASH OUT PAID TO DATE (EXPENDED)	PO BALANCE (OR ESTIMATE TO COMPLETE)	TOTAL PROJECT COST FORECAST
PRODUCT CATEGORY											
TOP LEVEL ASSEMBLY											
TOP LEVEL ASSEMBLY											
TOP LEVEL ASSEMBLY											
TOP LEVEL ASSEMBLY											
COIL HANDLING											
DOWNENDER/UPENDER											
COIL CAR											
UNCOILER											
LUBRICATOR											
PINCH ROLL STAND											
CONTROL PANEL											
REMOTE MIXING TANK											
OPERATOR CONSOLE											
SPARE PARTS											
SHEET FEED											
LUBRICATOR											
CONTROL PANEL											
REMOTE MIXING TANK											
OPERATOR CONSOLE											
SPARE PARTS											
CUP SYSTEM											
CUP PRESS											
ROLL FEED											
SCRAP CHOPPER											
SCRAP HOOD											
CONTROL PANEL											
COMPRESSED AIR INTERFACE											
COOLING WATER INTERFACE											
DIE SET											
OPERATOR CONSOLE											
SPARE PARTS											

Appendix D: "System Specifications Spreadsheet"

Step #	PROJECTS: Process users go through to establish and manage a project	Budget System Functions (what the system needs to do)	Data required for budget (needed to set up DB)	Source for budget data	Report or Forms for Budget
1 Set up Project					
1.1	User requests new budget	System must grant access based on user's authorization	Username and password	User entered, system stored, admin grants access	Request new project form Dave's Summary Report Template
		Update database fields with info needed to establish and track Project	Project manager	User entered	
		System checks that project does not already exist in QuickBooks	Project Name / ID	User entered	
			Project start date	User entered	
			Target delivery date	User entered	
			Project Standards		
			Speed: MXI, MXII, or MXIII	User entered	
			Material: Aluminium or Steel	User entered	
			Package: Food, Beverage, Bottle, Aerosol	User entered	
			Parameters: Height, Diameter, Profile	User entered	
			Project duration	Calculated =Target End Date - Start Date =Today - Start Date	
1.2	Project approved	System must save requested budget Lock the start date, project manager, project name/ID fields --- Where? Form?	Data Start date, project manager, project name/ID	From the request new project form Info as approved for baseline	
		Update QB from Solution	Project Name, manager (custom field), start date, projected end date, etc.)	Info as approved for baseline	
1.3	Users update project as more information becomes available	Display "Success!" or "Error" System must retrieve current data for project	Project manager, Project Name/ID, Start Date, Target End Date, Project Standards, Duration, etc.	System	Edit Project Form
2 Set up Budget / Cash Flow					
					Forms to enter initial budget Contract data entry form Draft Budget
2.1	Revenue / Cash-in	Enter Revenue Total into DB	Total Revenue for Project	From Contract	
2.2	Cost / Cash-out	Build DB fields to match Equipment categories in project and enter cost in each category.	Equipment Categories	From Contract - but ultimately from Project BOM	
			Budgeted Cost for Project Equipment Categories	From project manager - but ultimately from Project BOM	
		Direct cost categories (travel, contractors, etc.)			
		OII Services and Fixed Costs (Salary Employees, Buildings, Equipment/Machinery, etc.)			

Step #	PROJECTS: Process users go through to establish and manage a project	Budget System Functions (what the system needs to do)	Data required for budget (needed to set up DB)	Source for budget data	Report or Forms for Budget
3 Approve/set baseline budget					Baseline Budget
3.1	Approve budget	Confirm budget approval Store baseline budget, do not allow changes	Budget Approved (yes/no)	Project manager	
3.2	Sign Contract	Lock fields so baseline budget cannot be changed (except through change forms with proper authorization)	Contract Signed (yes/no)	Project manager	
4 Review Budget and Cash Flow					Cost Status Summary Report
					Ad Hoc Report
4.1	Authorize	Verify authorization to review budget and cash flow	Username and Password	User entered, system stored, admin grants access	
4.2	Select Standard Report	Present user with report selection options (Drop-downs, links, menus, etc)	Report Selected	User entered	Form for Standard Report Selection
4.3	Create Ad Hoc Report	Create Ad Hoc report based on user preferences and give option to save report for future access	Report Specifications	User entered	Wizard for Ad Hoc Report
4.4	Review Reports	Query for up-to-date data for report (for calcs and display)	Cost	QuickBooks	
			Revenue	QuickBooks	
			Delivery, Installation --- maybe (implicit through updated cost and rev)	System	
			Report Selected	User entered	
			Equipment Categories	Up-to-date budget	
		Display report to screen, print report, and allow report to be emailed and faxed	Report Selected	User entered	
4.5	Change Order	Modify the budget after change order is filled out, save as most up-to-date report data --- keep old data or replace?	Budgeted cost, budgeted revenue	Change Order Form	Change Order Form
5 Cash-In Functions					Cost Status Summary Report
5.1	Issue Invoice	N/A (occurs in QB)			
5.2	Receive Payment	Update total payments and calculate balance due	Equipment Categories	From up-to-date budget	
			Payment Receipt Date	From QB	
			Payment Amount	From QB	
5.3	Change Order - New revenue or Credit	Change budget	Equipment Categories	From Change Order Form	Change Order Form?? - or are change in QB good enough?
			Amount	From Change Order Form or QB?	
			Invoice date of Chg Order	From QB	
	Change Terms	N/A			

Step #	PROJECTS: Process users go through to establish and manage a project	Budget System Functions (what the system needs to do)	Data required for budget (needed to set up DB)	Source for budget data	Report or Forms for Budget
6 Cash-Out Functions					
6.1	Issue PO	N/A (occurs in QB)			
6.2	Receive Invoice(s)	N/A			
6.3	Receive Item	N/A			
6.4	Make Payment	Update actual cash-out based on payment made	Vendor used Check detail (actuals) Equipment Categories Purchase commitment	From QB From QB From QB	
6.5	Change Order - New cost or debit	Change baseline budget numbers and save as up-to-date budget	New Cost or Debit Equipment Categories	Change Order Form	Change Order Form
7 Cash-In/Out Trigger Events					
7.1	<u>Cash-in forecasted</u>	N/A			Data entry form (including anticipated ship date)
7.1.1	No invoice sent				
7.1.2	No payment received				
7.1.3	Partial payment received				
7.2	<u>Cash-out forecasted</u>				
7.2.1	No Item Received	Store anticipated ship date and allow changes to be made in system	Anticipated ship date	From Mike or Roy (System)	
7.2.2	Cash-out forecasted - No Invoice Received				
	>More				

Step #	PROJECTS: Process users go through to establish and manage a project	Cash Flow System Functions (what the system needs to do)	Data required for cash flow (needed to set up DB)	Source for cash flow data	Report or Forms for Cash Flows
3	Approve/set baseline budget				
3.1	Approve budget				
3.2	Sign Contract	Lock fields	Contract signed (yes/no)	Project manager	
4	Review Budget and Cash Flow				Cash Flow Report
4.1	Authorize	Verify authorization to review budget and cash flow	Username and Password	User entered, system stored, admin grants access	Ad Hoc Reports
4.2	Select Standard Report	Present user with report selection options	Report Selected	User entered	Form for Standard Report Selection
4.3	Create Ad Hoc Report	Create Ad Hoc report based on user preferences and give option to save report for future access	Report Specifications	User entered	Wizard for Ad Hoc Report
4.4	Review Reports	Query for up-to-date data for report (for calcs and display)	Cost Revenue Delivery, Installation --- maybe (implicit through updated cost and rev) Report Selected	QuickBooks QuickBooks System User entered	
		Display report to screen, print report, and allow report to be emailed and faxed	Report Selected	User entered	
4.5	Change Order	Change order updates Purchasing Schedule and/or Cash-in data, which System uses to recalculate cash flow Modify the cash flow with updates from Purchasing Schedule and Cash-in (from Chg Order Fm), save as most up-to-date report data --- keep old data or replace?	Purchasing Schedule Update, cash-in payment terms Lead time, terms	Change Order Form Purchasing Schedule, Change Order Form --- (make a Receiving Schedule)	Change Order Form
5	Cash-In Functions				
5.1	Issue Invoice	N/A (occurs in QB) Update cash flow based on invoice date and other triggers	Invoice number, date	From QB	Monthly Cash Flow Report
5.2	Receive Payment	Track payments that have been received, update the forecasted/actual cash flow If amount = payment forecasted = OK If amount > payment forecasted = deduct excess from last payment If amount < payment forecasted = add difference to next payment	Equipment Categories Payment Receipt Date Payment Amount Payment Forecasted	From up-to-date budget From QB From QB From QB	
5.3	Change Order - New revenue or Credit	Update cash-in	Equipment Categories Amount Terms for payment receipt Invoice date of Chg Order	From Change Order Form (COF) From Change Order Form or QB From QB or COF From QB	Change Order Form?? - or are change in QB good enough?
	Change Terms	Change terms going forward and recalculate cash flow forecast	New Terms for Payments	From Terms Change Form	Terms Change Form

Step #	PROJECTS: Process users go through to establish and manage a project	Cash Flow System Functions (what the system needs to do)	Data required for cash flow (needed to set up DB)	Source for cash flow data	Report or Forms for Cash Flows
6 Cash-Out Functions					
6.1	Issue PO	N/A (occurs in QB) Make cash flow calculations based on defaults from PO, update calculations as further data is acquired	PO Issue Date	From QB	Monthly Cash Flow Report
			PO Amount	From QB	
			Cash-out payment terms	From QB	
6.2	Receive Invoice(s)	Invoice triggers update in Payment Schedule (e.g. first invoice triggers down payment 10 days after date invoice is received)	Invoice Date	From QB	
			Invoice Amount	From QB	
			Cash-out payment terms	From QB	
			Equipment Categories	From Invoice - user entered or QB	
			Purchase commitment	From QB	
6.3	Receive Item	Delivery date triggers update in Payment Schedule	Delivery Date	User entered	
			Equipment Categories	From user	
			Cash-out payment terms	From QB	
6.4	Make Payment	Update cash flow with actuals	Check detail (actuals)	From QB	
6.5	Change Order - New cost or debit	Update forecast with new cost or debit			
7 Cash-In/Out Trigger Events					
7.1 Cash-in forecasted					
7.1.1	No invoice sent	Store expected invoice sent date	Invoice sent or expected invoice sent date	From QB	Data entry form (including ship date, installation date)
7.1.2	No payment received	Calculate expected payment receipt date	Expected payment received date or actual date received	From QB or calcs	
7.1.3	Partial payment received	Store payment received and calculate remainder of payments	Actual payment received	From QB	
7.2 Cash-out forecasted					
7.2.1	No Item Received	Calculate expected item arrival	Expected item receipt date or actual receipt date	From QB or Mike	
7.2.2	Cash-out forecasted - No Invoice Received	Calculate expected invoice arrival and calculate expected cash-out date based on PO issue date	Invoice received date, payment made date	From QB	
	>More		PO date	From QB	
			Ship date	From Mike (System)	
			Installation date	From Mike (System)	

Appendix E: CSSR Code in PHP

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>

    <?php

#####
# The code below is used to extract data from quickbooks and display it in a table so it looks similar to the project
# manager's current Cost Status Summary Report
#####

$projectName = $_GET['projectName'];
$startDate = $_GET['startDate'];
$endDate = $_GET['endDate'];

#Connect to a System DSN "QuickBooks Data" with no user or password
$soConnect = odbc_connect("QuickBooks Data", "", "");

function showCategory($categoryName)
{
    global $soConnect;
    global $projectName;
    global $startDate;
    global $endDate;

    echo "<h2>$categoryName</h2>
        <table border='0' border='0' bgcolor='#888888' bordercolor='black' cellpadding='0' cellspacing='0'>
            <tr>
                <td>
                    <table border='1' bordercolor='black' bgcolor='white' cellpadding='5' cellspacing='0'>
                        <tr>
                            <!--<th>Row</th-->";

    set_time_limit(120);

#Set the SQL Statement
$sSQL = "SELECT PurchaseOrderLineItemRefFullName AS DescriptionOfEquipment, VendorRefFullName,
    TxnDate, TermsRefFullName, ExpectedDate, PurchaseOrderLineAmount
    FROM PurchaseOrderLine
    WHERE (TxnDate >= {d'$startDate'}) AND (TxnDate <= {d'$endDate'})
    AND ((PurchaseOrderLineItemRefFullName Like '%'. $categoryName . '%')
    OR (PurchaseOrderLineItemRefFullName Like '%'. $categoryName . ':%'))
    AND (PurchaseOrderLineCustomerRefFullName Like '%'. $projectName . '%')
    ORDER BY PurchaseOrderLineCustomerRefFullName, TxnDate";

#echo $sSQL;

#Perform the query
$soResult = odbc_exec($soConnect, $sSQL);
    print("                <th>Description of Equipment</th>\n");
    print("                <th>Vendor Selected</th>\n");
    print("                <th>PO Issue Date</th>\n");
    print("                <th>Payment Terms</th>\n");
    print("                <th>Anticipated Ship Date</th>\n");
    print("                <th>Contract Project Value <em>(Prorata)</em></th>\n");
    print("                <th>Project Budget <em>(Estimated Cost)</em></th>\n");
    print("                <th>Purchase Commitment <em>(Actuals)</em></th>\n");
    print("                <th>Cash Out Paid To Date <em>(Expended)</em></th>\n");
    print("                <th>PO Balance (Or Estimate to Complete)</th>\n");
    print("                <th>Total Project Cost Forecast</th>\n");
    print("            </tr>\n");
    print("        </thead>\n");
    print("        <tbody>\n");
    $iRecCnt = 0;
#Fetch the data from the database
while($row = odbc_fetch_array($soResult)) {
    $iRecCnt++;
    print("                <tr>\n");
    ##
        If ($row['DescriptionOfEquipment'] == "") {
```

```

        print("                <td> </td>\n");
    }
    else {
        print("                <td valign="top">" . $row['DescriptionOfEquipment'] . "</td>\n");
    }
}
##
If ($row['VendorRefFullName'] == "") {
    print("                <td> </td>\n");
}
else {
    print("                <td valign="top">" . $row['VendorRefFullName'] . "</td>\n");
}
##
If ($row['TxnDate'] == "") {
    print("                <td> </td>\n");
}
else {
    print("                <td valign="top">" . $row['TxnDate'] . "</td>\n");
}
##
If ($row['TermsRefFullName'] == "") {
    print("                <td> </td>\n");
}
else {
    print("                <td valign="top">" . $row['TermsRefFullName'] . "</td>\n");
}
##
If ($row['ExpectedDate'] == "") {
    print("                <td> </td>\n");
}
else {
    print("                <td valign="top">" . $row['ExpectedDate'] . "</td>\n");
}
print("                <td align="center" valign="center"><input align="right"
id="" . $lRecCnt . $categoryName . "ContractProjectValue" name="" . $lRecCnt . $categoryName . "ContractProjectValue"
type="text" size="10" value="0.00"/></td>\n"); #Contract Project Value <em>(Prorata)</em>
print("                <td align="center" valign="center"><input align="right"
id="" . $lRecCnt . $categoryName . "ProjectBudget" name="" . $lRecCnt . $categoryName . "ProjectBudget" type="text"
size="10" value="0.00"/></td>\n"); #Project Budget <em>(Estimated Cost)</em>
print("                <td align="center" valign="center"><input align="right"
id="" . $lRecCnt . $categoryName . "PurchaseOrderLineAmount"
name="" . $lRecCnt . $categoryName . "PurchaseOrderLineAmount" type="text" size="10"
value="" . number_format($row['PurchaseOrderLineAmount'], 2, ',', '') . "</td>\n"); #Purchase
Commitment <em>(Actuals)</em>
print("                <td align="center" valign="center"><input
onblur="computePOBalance(' . $lRecCnt . $categoryName . "PurchaseOrderLineAmount', this.id,
' . $lRecCnt . $categoryName . "POBalance')\" align="right" id="" . $lRecCnt . $categoryName . "CashOut"
name="" . $lRecCnt . $categoryName . "CashOut" type="text" size="10" value="0.00" /></td>\n");
print("                <td align="center" valign="center"><input align="right"
id="" . $lRecCnt . $categoryName . "POBalance" name="" . $lRecCnt . $categoryName . "POBalance" type="text" size="10"
value="0.00" disabled="disabled" /></td>\n");
print("                <td align="center" valign="center"><input align="right"
id="" . $lRecCnt . $categoryName . "CostForecast" name="" . $lRecCnt . $categoryName . "CostForecast" type="text"
size="10" value="0.00" disabled="disabled" /></td>\n");
print("                </tr>\n");
    }
}
echo "                </tbody>
                </table>
            </td>
        </tr>
    </table>";
}
?>
<title><?php echo $projectName; ?> Cost Status Summary Report</title>

<style>
body{
margin: 10px 0px 10px 10px;
background: #eeeeee;
font-size: small;
font-family: Tahoma, Arial, "Trebuchet MS", Helvetica, sans-serif;
text-align: left;
}

table{
margin: 5px 5px 5px 5px;
}

```

```

.cssform {
    padding: 10px 0px 0px 10px;
    font-size: 100%;
}

.cssform p {
width: 300px;
margin: 0;
padding: 5px 0 8px 0;
padding-left: 155px; /*width of left column containing the label elements*/
height: 1%;
}

.cssform label {
font-weight: bold;
float: left;
margin-left: -150px; /*width of left column*/
width: 150px; /*width of labels. Should be smaller than left column (155px) to create some right margin*/
color: #334d55;
}

.cssform input[type="text"] { /*width of text boxes. IE6 does not understand this attribute*/
width: 180px;
}

.cssform textarea {
width: 250px;
height: 150px;
}
</style>

<script type="text/javascript">
function computePOBalance(commitment, cashOut, poBalance)
{
    var myCommitment=document.getElementById(commitment).value;
    var myCashOut=document.getElementById(cashOut).value;
    document.getElementById(poBalance).value=(parseFloat(myCommitment)-parseFloat(myCashOut)).toFixed(2);
    document.getElementById(cashOut).value=(parseFloat(myCashOut)).toFixed(2);
}
</script>
</head>
<body topmargin="3" leftmargin="3" marginheight="0" marginwidth="0" bgcolor="#ffffff" link="#000066" vlink="#000000" alink="#0000ff" text="#000000">
<h1><?php echo $projectName; ?> Cost Status Summary Report </h1>
<!--
-----
The form below allows users to chose which project they would like to see data for, as well as beginning and end dates
-----
-->
<form action="CSSR.php" method="get" class="cssform" name="SelectReport" target="_self">
<p>
<label>Start Date:</label>
<input id="startDate" name="startDate" type="text" value="1995-01-01"><br />
</p>

<p>
<label>End Date:</label>
<input id="endDate" name="endDate" type="text" value="<?php echo date("Y-m-d"); ?>"><br />
</p>

<p>
<label>Project:</label>
<select id="projectName" name="projectName">
<option>OCS1</option>
<option selected="selected">OCS2</option>
<option>MX0003</option>
</select>
<input type="submit" value="Submit">
</p>
</form>
<?php
    $sSQL = "SELECT Name FROM Account WHERE AccountNumber LIKE '5010%' AND AccountNumber NOT LIKE
'5010'";
    $oResult = odbc_exec($oConnect, $sSQL);
    while($row = odbc_fetch_array($oResult)) {
        showCategory($row['Name']);
    }

```

```
?> }  
</body>  
</html>
```