Table Of Contents

Overview

A discussion of this type can be expanded into volumes. I have tried to include those issues germane to the topic of upgrading existing Basic2c/NPL applications. Obviously the contents are based on my experiences, and will be biased to that extent, however, I believe this document to be the basis for an open discussion of the issues. And, should be enhanced in a collaborative effort to fully define the development universe in which we find ourselves now and into the foreseeable future.

I agree that if pursued, this will be as important as the decision to write BASIC2C/NPL and will not only provide the VARs with development products needed to upgrade legacy applications, but also for the creation of the new generation of business applications that are being demanded by business owners world wide.

S. Matzen.

Challenge

Creating a tool suite that will allow developers to modernize their existing software products is an interesting challenge. The following criterion makes for an excellent starting point:

1. Connect their NPL application to the Internet (or intranet, or network, or operate standalone) using a browser interface.
2. Net-enable their entire (potentially huge) application (including browser interface and transaction processing) in under a week.
3. Do so in 1999.

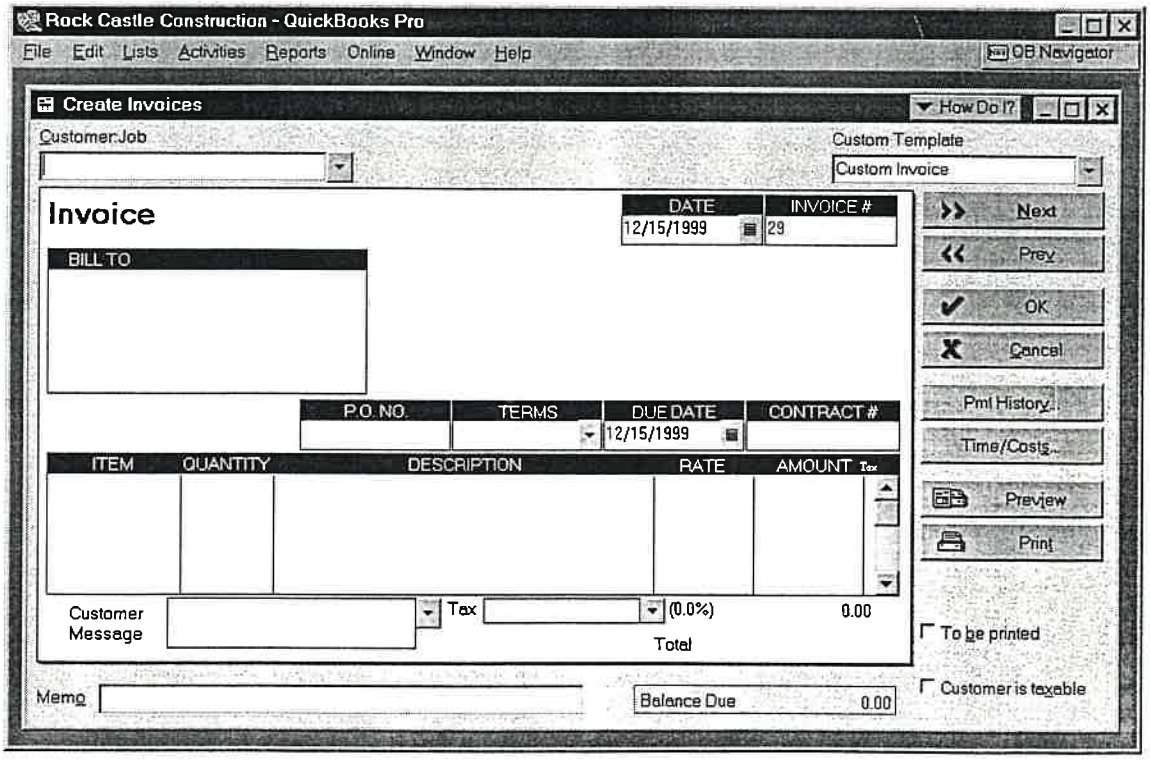
Modern User Interface Presentations

There are two entirely different user interface presentations deployed in the modern environment and three sub-types that must be considered:

Windows

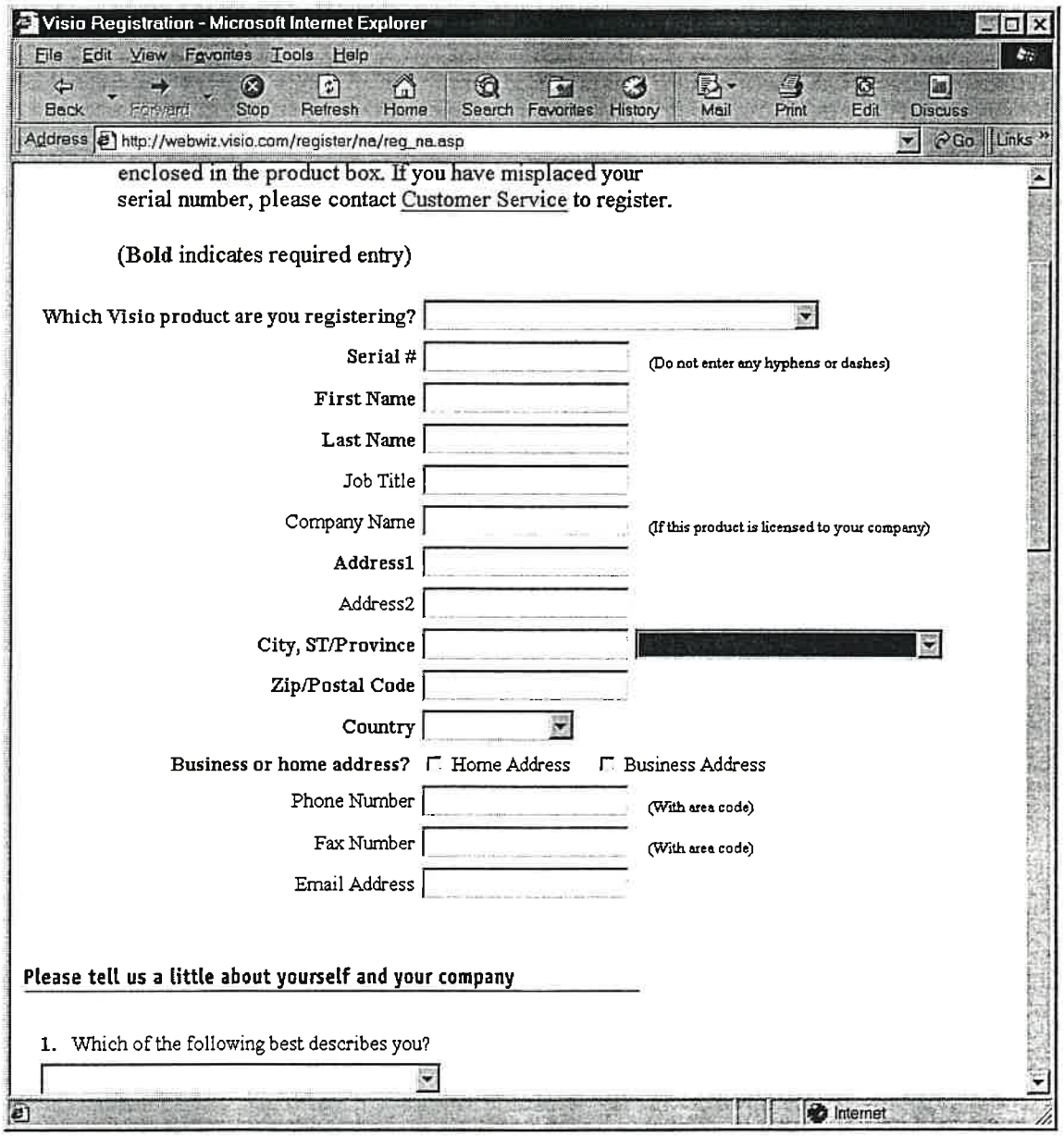
Windows -- Form based input with almost unlimited display control options including combo boxes, radio buttons, control buttons, labels and text boxes, and many kinds of grids. Flow control is handled with drop-down menus and button bars. There is typically a main program that contains multiple forms (documents) called a multi-document-interface, thus multiple forms can be active at one time. Within the Windows interfaces there are three different end user experience levels to consider

1. **Consumer** -- Home users and consumers with limited computer experience. Microsoft Money is a good example of a consumer product.
2. **User** -- Experienced computer user. These are users, as we now know them.
3. **Managers** -- Inexperienced users that require special interfaces for the presentation of summary management data with links details presented in a management reporting style.



Web

Web -- Form based input with limited display control options. Most web applications provide not-only form input, but informational content. A Typical e-commerce site provides the user primarily with information content and uses form based input to extract information from the user. The flow of a **Web** application is vastly different than the flow of a **Windows** application. The application must be designed to work on different web browsers and widely varying display geometry.



Note: A typical application provides a complete **Windows** interface and a limited number of **Web** forms.

User Interfaces

Different applications require different user interfaces. Currently these interfaces need to be considered:

- Microsoft Windows on a workstation (pc)
- Microsoft Windows on a Windows Terminal
- HTML based web applications
- Windows Terminal launched by a web browser
- Character terminal
- Character terminal emulator on a workstation
- Graphical interface emulation of character based application
- X Windows on Unix
- Mac Windows on Mackintosh

Modern User Interface Behavior

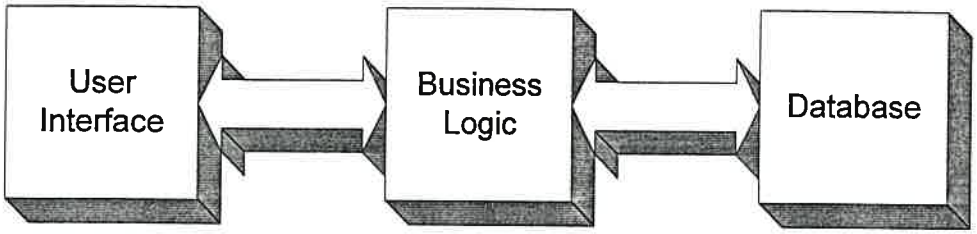
Modern application user interfaces exhibit a number of behaviors that are foreign to legacy BASIC2/NPL applications:

- **Menu structure and program flow control** are an integral part of the user interface (windows). Most legacy applications have some form of menu system for program selection, with additional program flow control implemented in the business logic.
- **Event processing**. The windows environment requires that the user be able to move the cursor from one input field (text box) to another with the mouse. Many legacy applications have program logic that leads the user from one field to another.
- **Transaction processing**. The web interface is generally a transaction processing style interface where the user enters data into a form and submits the form for processing. Most legacy applications process data field-by-field.
- **Reporting**. Most reporting requires some form of user interface for report selection entry and subsequently the report is prepared and printed. Separating the user interface logic from the report logic can be difficult with some legacy applications.

The Whole Picture

I believe that we need to study the whole picture before we start trying to solve the problem. In other words, let's take a detailed view of the business application components required to deploy in a modern environment:

- User Interface
- Business Logic
- Database



User Interface Component

There are typically two primary user interface classifications:

Display

In the display interface, information is displayed on some form of display and the operator responds interactively with the display with a keyboard, pointing device, voice, or other human/computer interface.

There are three broad categories of display interfaces into-which each of the currently used interfaces can be placed:

Character

Character terminal

Character terminal emulator on a workstation

Graphical interface emulation of character based application

Windows

Microsoft Windows on a workstation (a workstation is a pc)

Microsoft Windows on a Windows Terminal

Windows Terminal launched by a web browser

X Windows on Unix

Mac Windows on Mackintosh

Web

HTML based web applications

Report

The report interface allows the user to select what is to be printed, and the print details are processed through a printer driver and subsequently a printed document is prepared.

There are two categories of report interface:

Character

These are reports printed on character printers. Generally they are of a single type style, character width and lines per page

Graphical

These are reports that include variable type styles, lines, graphics and pictures.

Business Logic Component

The business component includes all program logic used to provide a bridge between the **user interface** and the **database.**

Database Component

The database component includes the storage and retrieval of persistent data.

Solutions

I have studied these issues for the past few years and have determined that there are three approaches to modernizing a legacy BASIC2C/NPL application:

Graphical Interface Emulation

Simply stated, it maintains the existing application with little or no modifications and puts a more modern presentation on the fields. By using the 16 attributes, an interface emulator will capture the labels and text boxes and display them on the windows screen in standard windows format. The developer/user will be allowed to modify the presentation styles and colors, but the application will not change. The menus, data entry screens, reporting and database will not change.

For the **Unix** based applications using dumb terminals or terminal emulators on personal workstations, deployment of a graphical interface emulator will be quite easy. A web browser can launch this interface emulator, thus the application can be immediately web enabled.

For the **Windows** based applications, we will probably need to provide a tight interface between the graphical interface emulation and the windows NPL runtime, a relatively simple task since the runtime already supports remote control. At this time the only way I know of getting these applications web-enabled is to put them on Unix where we launch the application from a web browser. There may be a way to make this happen on NT, but I am not familiar with it.

Up Side

- Can be deployed quickly.
- The existing application will need minimal to no changes.

Down Side

- This approach has been implemented by other companies and has met with limited success.
- Latency may be an issue for those applications deployed on the web. The graphical interface emulator leaves all of the logic with the existing application. Since this solution doesn't modify the application logic (which includes the user interface), each character must be sent to the application, processed and returned to the user interface, thus latency. Typical latency with a 56k connection will be in the 100ms range. For some applications this is acceptable, for some it isn't.

- This will not provide a web page-style block-mode interface.

Workbench Assisted Application Enhancement

This approach would have the NPL Workbench parse an existing application in both static and active (while it is being run) and modify the existing application code to be more modern. This would require that the workbench be able to convert the existing procedural field-by-field applications into form-based applications. Form-based applications allow the user to enter all data on a form and then submit the form. Field relationship editing is limited and many of the display elements would need to be modified to work with the modern environment.

Program logic would have to be modified to work with the form paradigm, and for many of the existing legacy applications, these modifications would have to be extensive.

A graphical workbench tool would need to be available to modify the display layout. The existing display real estate of 80x24 or 132x24 works well for character interfaces, but will need to be reworked for the graphical interface, especially the web interface where the user can change the display geometry.

Up Side

- Existing report programs would not require much modification. The user interface would be extracted into a form and the report would remain essentially the same.
- Much of the existing application logic can be preserved and the database won't have to be touched.
- The new application will be able to coexist with the existing character based application.

Down Side

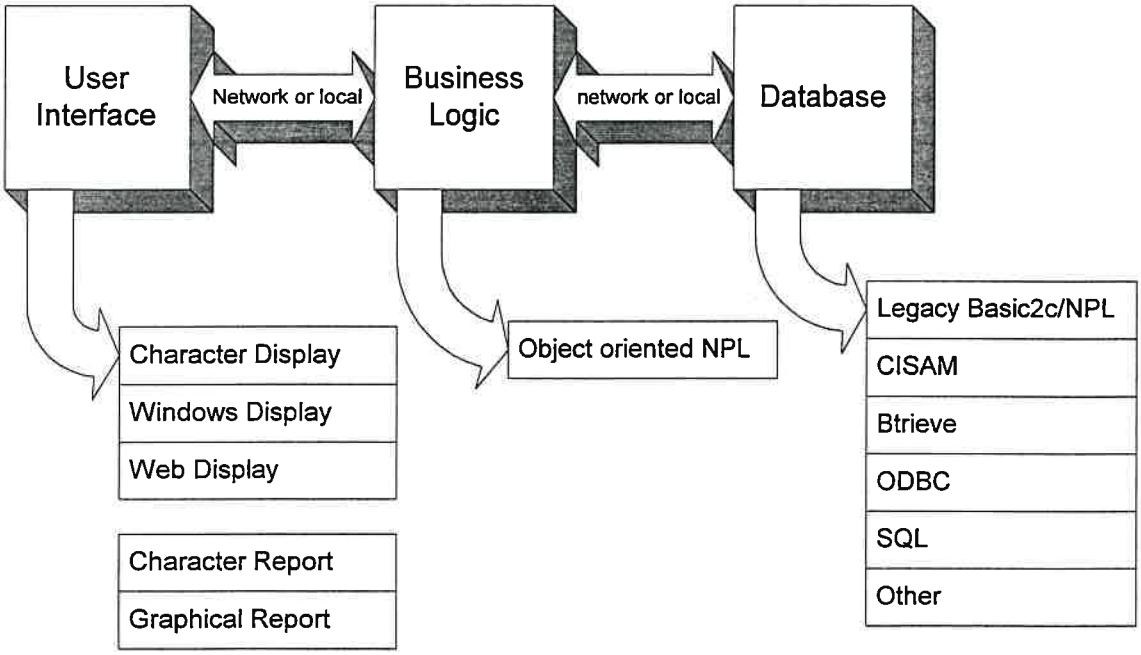
- The workbench tools can't do it all. The developer will have to write some business logic to supply different display controls (combo boxes, radio buttons, ...) with the necessary data tables and values. The most-used programs (generally 3 to 5 in each application) will probably have to be completely re-constructed to work with the form paradigm.
- The menus and flow control will either have to be re-written or provide a non-standard presentation in the windows environment.
- The presentation won't be full windows compliant.
- After all this work, the application will probably have to be re-deployed anyway.

Application Re-deployment

Through enhancements to the NPL Workbench and Integrated Development Environment, the application is deployed in a true multi-tiered-network environment that separates the user interface, business logic and database interface into three separate areas.

- Application will operate under industry standard window managers including Microsoft Windows.
- Resultant system will be structured so that it can be merged into a web based informational interface as an e-commerce application.
- Will provide database interface for transaction processing.

This will be accomplished by providing the developer with the tools necessary to quickly deploy an application into this highly advanced environment:



Assuming the business application is just being re-deployed and not redesigned and deployed, the deployment process will include the following steps:

1. Use the workbench to model the application including the links between the user interface forms and reports. This modeling provides the basis for generating the initial windows with menu options to provide for transfer from one window to another.

2. Use the workbench to model the database. A tool will be made available to extract much of the existing database model from the legacy application.
3. Use the workbench to model the database constraints. A tool will be made available to extract many of the existing constraints from the legacy application. Constraints are the basis for input verification and provide the details necessary for creation of select and combo boxes.
4. Use the workbench to model the database relationships.
5. Use the workbench to model the business logic classes. This includes modeling the user interface and database interface properties, events and methods.
6. Use the workbench to model the web application forms.
7. Use the workbench to model each individual form and report. A tool will be made available to extract much of the content from the legacy application.
8. Use the workbench to generate the application framework.
9. Use the workbench to enhance, modify and test the application prior to test deployment.
10. Use the workbench to model the data conversion and generate the data conversion programs. A tool will be made available to extract much of the data conversion details from the legacy application.
11. Deploy the application in test environments and use the workbench to make modifications. Use the workbench update deployment tool to deploy updates into the text environments.
12. Deploy the application into the production environment.

Much of the success of this type of re-deployment depends on the amount of application modification being made during re-deployment. Because of the restrictive environment in which many of the legacy environments were developed, there may be a long list of enhancements that will be made during re-deployment.

Up Side

- Resultant application will be able to effectively deploy on all classes of user interface including windows, web and character.
- Resultant application will be able to deploy in the heterogeneous computing environment including Windows 95/98, Windows 2000, Windows NT, and the Intel Unix variants.
- Resultant application will be cross-platform deployable.

- Resultant application will be web enabled.
- Resultant application will be modern, thus reducing the reluctance of new developers to enter the NPL arena.
- Completely standards compliant modern look and feel.
- Niakwa will have an enterprise class application development product to sell.

Down Side

- Will require significant enhancements to workbench.
- Will require some VAR training.

Can We Meet The Stated Objectives?

I believe that we can provide some relief to the community by providing the **Graphical Interface Emulation** solution. It isn't for everyone, but may provide some relief for those VARs that are under a lot of pressure and aren't sure what to do.

The real solution is to take a leap over the current technology and land in the future where we provide the VARs with a tool set that will let them **re-deploy** legacy applications and **deploy new** applications using modern technology. Integrating Visual NPL, Open NDM, and Visual NDM, into the workbench and enhancing the workbench with a strong tool set will meet both the short and long term needs of the Niakwa VARs.

Can we connect their NPL applications to the Internet using a browser interface? This is highly application dependent, and will depend on what types of data are to be extracted from the web interface. For most applications it is not practical (or desirable) to make all components applicable to the web. However, all components can be deployed in Windows and a few of the consumer class forms implemented on the web.

Can we net-enable their entire (potentially huge) application (including browser interface and transaction processing) in under a week? Probably not. There isn't enough business logic in the current applications to make this happen. Most applications don't lend themselves well to transaction and/or forms based processing. What we can do is provide the developer with tools to re-deploy his application on top of a template that will allow for all types of user interfaces, isolate his business logic, and provide access to unlimited types of databases.

Can we do this in 1999? If we don't let first release of the project creep very much we can ship in 1999. This will require some quick decision making on your part. If you wait until July, it isn't going to happen. However, I have been working on this problem for about 5 years now, and have many of the components in some form of completion and deployment. So, we won't be starting from scratch.

How Do We Get There?

1. We get the development team together (no more than three people) for about a week to layout the complete project. That team then decides what components can be completed in the given time frame.
2. We detail plan the project. (A lot like we did with the workbench, but with a little more detail.)
3. We get sample code from the VARs that represent the typical applications to be re-deployed.
4. We write the workbench enhancements and tools necessary to meet the project goals.

What Else Is There?

Right now I can think of about 25 more issues that need to be discussed, but this document is as long as it needs to be already.

If you would like to pursue this direction, let me know and we will figure out how to get it done.

S. Matzen