MOVING THE SENIOR DEVELOPMENT CLASS FROM WEB DEVELOPMENT TO LIFE CYCLE DEVELOPMENT – A CASE FOR VISUAL STUDIO 2005

Thom Luce, Ohio University, luce@ohio.edu

ABSTRACT

Information Systems programs in Business Colleges are often constrained in the number of classes they can offer majors by both university and accreditation rules. Some programs attempt to get around these constraints by combining things like application and web page development tools and techniques in a single class with broader system development and project management concepts. While this may work in a semester based system, it can severely push the limits of what is possible for quarter based programs. This paper describes the evolution and current, or near future, state of one possible solution to this problem, the use of Microsoft's ASP as a development tool. The paper focuses on features in the forthcoming ASP.NET 2.0 and Visual Studio 2005 that simplify the development/programming environment while permitting students to create more sophisticated, more robust applications in less time than was previously possible. The time saved on development efforts can provide students with more time for requirements analysis, risk assessment, testing, documentation, project management and other aspects of the full development cycle.

Keywords: System Development, ASP.NET, Visual Studio, Web Development, Teaching Tools

INTRODUCTION

Microsoft's release of Active Server Pages (ASP) in the late 1990's changed many college and university systems development classes. ASP provided a relatively cheap, relatively easy to use tool with which students could develop interactive web applications in a Microsoft environment. ASP ran on Microsoft's Internet Information Server (IIS) version 3.0 and above and Microsoft's Personal Web Server (PWS) thus allowing students to develop and test web applications on their personal computers. Server-side programming in ASP was typically done with VBScript while client-side programming was in JavaScript. Server-side code was mixed with client-side code and standard HTML via special script tags. ASP worked with Microsoft's ActiveX Data Object (ADO) to provide simple recordset access to server-side databases (Microsoft Access, SQL Server, Oracle, etc.) [1].

Schools such as Ohio University introduced ASP as part of the first systems development class and then used it as a tool in the second class. Other schools taught ASP as part of a separate web development class. The approach used at Ohio University assumed that students with relatively little programming background (a quarter of Visual Basic and a quarter of Java) would be able to take the basics of ASP presented in the first class and build on them as needed in the second class.

Students at Ohio University were able to progress much further in a quarter than was previously possible but they rarely developed as far as hoped. Some of the limiting factors included:

- Confusion caused by source pages with mixed client-side and server-side code
- Problems associated with the validation of user input
- Difficulties repopulating web forms with user supplied values when errors were discovered
- Problems associated with how a page was accessed (i.e., new access via url, post back, redirect, etc.)
- Data access problems related to limitations in the recordset model
- Login control
- Consistent look-and-feel
- Navigation

Also, regardless of the reduced development time afforded by ASP, students taking classes using the quarter system still had little time for other aspects of the systems development cycle or its management.

Microsoft addressed many of these problems with the release of the .NET Framework, ASP.NET 1.0 and the subsequent ASP.NET 1.1. First, the programming model changed to allow web page development in any of the .NET languages including VB.NET and the new C#.NET. Second, code was moved from the web page to a separate "code behind" page thus eliminating confusion caused by mixing HTML and programming on the same page. This approach also promoted the n-tier architectural separation of user interface and business logic.

ASP.NET 1.0 introduced a series of server-side controls that extended the functionality of the original HTML controls. ASP user interface controls (asp:Label, asp:TextBox, asp:RadioButton, etc.) brought a large number of properties and events to the corresponding HTML control. Many of the properties are ultimately converted to cascading style sheet settings associated with standard HTML tags. Because ASP controls "run-at" the server, they have properties and methods that can be modified and used as the page is processed at the server. Most ASP controls have a PostBack property that permits the control to generate a post back event as soon as something happens at the user interface (a button is clicked, the user selects a new item from a drop down list, a radio button is pressed, etc.) rather than waiting for the entire form to be posted to the server. [2]

ASP.NET introduced a series of validator controls (required field, range, compare, regular expression and custom) that simplify the validation process, often eliminating the need for the student to write any validation code. This, along with the introduction of the ViewState, makes it very easy to create and validate form data. All user interface ASP controls support the ViewState – an object that holds the current contents of the controls between post backs. Controls with the ViewState enabled are able to "remember" what they contain between postbacks and display previously entered data without program intervention.

Along with ASP.NET, the .NET framework came with a new version of ADO known as ADO.NET [3]. This replaced the old recordset with several new controls including a datareader and a dataset. In addition, a number of new bound controls, controls that can directly bind to and display data, were introduced. These include the dropdown list and listbox along with a new datagrid and a repeater control.

Visual Studio 2002/2003 offers a series of wizards that simplify the often complex tasks such as creating data connections and data adapters and mapping the resulting datasets to bound controls such as the datagrid. The visual interface provided by Visual Studio also simplifies the creation of single and multiple table database operations.

The combined effect of these and other changes is a marked increase in student productivity. Form creation and post back display require essentially no coding. Data validation is often handled automatically. Database operations and the display of results are both simplified and expanded from the original ASP/ADO models. The end result – students are more productive and can get much further into all aspects of the project than was possible with the pre-.NET product.

There are, however, some important problems. First, look-and-feel issues are lessened with the advent of the ViewState but there is still no easy way to assure a consistent look-and-feel over a series of pages. Also, while the data access controls help with problems such as login control, these areas still require coding and the student's use of session variables, cookies or some other state management system.

Additional problems occur because Visual Studio 2002/2003 and ASP.NET 1.0 were developed for professional developers, not a teaching lab facility. Transfer of ASP projects from the student's computer to a web server generally requires the presence of FrontPage Extensions on the server, something that many colleges and ISP don't support.

Debugging is another problem in Visual Studio 2002/2003. Visual Studio has built-in debugging options including the ability to run pages in debug mode, to set break points, to step into, over and out of code in breakpoint mode, along with the ability to examine and set any number of local variables. Unfortunately these features only work when the web application runs on the web server by a user with sufficient rights – a combination of requirements most students never meet.

ASP.NET 2.0 AND VISUAL STUDIO 2005

Microsoft is currently developing and testing ASP.NET 2.0 [4] and Visual Studio 2005, formally known as Visual Studio Whidbey and currently in the first beta release[5]. Early studies of the features and power of Visual Studio 2005 convinced the MIS Department at Ohio University to adapt it as the tool of choice for our system development and programming classes (VB.NET and C#.NET), even though it is still an early beta product. This paper describes some of those features and the rational for using the beta product.

First, Visual Studio 2005 has a built-in version of Internet Information Server that can run web pages (.aspx), web service (.asmx) and other ASP.NET 2.0 files on the local computer. This means that students can create and test web pages on a computer that doesn't have IIS installed and isn't otherwise a server. This internal version of IIS only runs on "localhost" so no external access or security problems are created by its use. This environment also supports all the debugging tools built into Visual Studio thus allowing the student to debug web pages faster and easier than was previously possible. Completed projects can generally be transferred to a server

with FTP, built into Visual Studio 2005, or a VPN connection and without the use of FrontPage Extensions.

Much of the functionality associated with server controls is now managed declaratively, rather then in code. As a very simple example, the button control in ASP.NET 1.1 is

```
<asp:Button id="Button1" runat="server" Text="Submit">
</asp:Button>
```

But it also requires a declaration on the code behind page

```
protected System.Web.UI.WebControls.Button Button1;
```

and a statement to wire the event handler to the button

```
this.Button1.Click += new System.EventHandler(this.Button1_Click);
```

The statements on the code behind page are automatically generated by Visual Studio when the a button is added to the form designer. Adding a button in Visual Studio 2005 creates the following code on the .aspx page

```
<asp:Button ID="Button1" Runat="server" Text="Submit"
OnClick="Button1_Click" />
```

the event handler is wired directly in the asp control and no addition code is created in the code behind page.

A much more extensive example is seen when an Access data source (new in ASP.NET 2.0 and discussed below) is added to a web page in Visual Studio 2005. All the code for the data source, including the select, insert, update and delete commands and all their parameters, is located in around twenty lines (depending on the complexity of the table) of code on the .aspx page. The corresponding operations in ASP.NET, creation of a data connection and data adapter, result in more than 100 lines of code on the code-behind page.

ASP.NET 2.0 now supports Master Pages which greatly reduce the time and effort necessary to create a consistent look and feel for a web site. Figure 1 shows the master page from an application being developed for IACIS. As shown in the diagram, a master page contains one or more content areas that will contain information when pages are developed. Figure 1 shows two such areas, cphBody, which will display the main body of page and cphMenu, which holds a menu.

Figure 1. A Master Page with Two Content Areas

Figure 3 illustrates a page created using the master page in Figure 1. The .aspx page itself contains the following Page directive indicating which master page to use:

```
<%@ Page Language="C#" MasterPageFile="~/IACIS.master" CompileWith =
"Submission.aspx.cs" ClassName="Submission_aspx" Title="Submission Page" %>
```

ASP.NET 2.0 simplifies and expands the data handling controls found in ASP.NET 1.1. Most data manipulation can now be handled with data sources (SqlDataSource, AccessDataSource, XmlDataSource, etc). Data sources functionally replace and extend the data connection and data adaptor objects found in ASP.NET 1.1 but those controls are still available for those who want or need them.

Figure 2 shows steps in the configuration of an AccessDataAdaptor. The top of the figure shows the data adaptor as it appears in design mode. The SmartTasks menu, another expanded feature in Visual Studio 2005, lists Configure Data Source as the only currently available option. Clicking Configure Data Source displays the configure window. From here you may select any table or query in the database along with the desired fields, or elect to create a custom SQL statement. The later selection displays a query editor window (not included in the figure). Once the table and fields are selected you may configure both the Order By and Where clauses of the SQL statement. Pressing the Where button causes another Window to open (shown at the bottom) where you specify any desired filtering and the location of parameters used in the Where clause. The example creates a Where clause that filters for Countries equal to the current contents of the txtCountry TextBox. Finally, pressing the Advanced Options button opens a window where you may select data modification and concurrency options. The first option will only be available if the current table or query is updatable.

In addition to the data source controls, ASP.NET 2.0 has a number of new bound controls including the GridView (a powerful replacement for the DataGrid), the DetailView and the FormView.

Figure 3 shows an example of a GridView (top) linked to a FormView (bottom). With the exception of three lines of code, all coding necessary to make these controls work is done in the .aspx page, not in the code-behind file. The exceptions are all commands that force the GridView to update its display after data is changed in the FormView.

In addition to the features illustrated in this paper, ASP.NET 2.0 now supports membership management through a series of controls that allow creation and maintenance of user ids, role assignment, login control, and content display based on login id and role. Additional controls support web site navigation with site maps (breadcrumbs) and menus (the TreeView control). ASP.NET 2.0 also supports personalization through the use of Themes and Skins and user input validation is simplified through the introduction of Validation Groups that allow logical grouping of controls and validators.

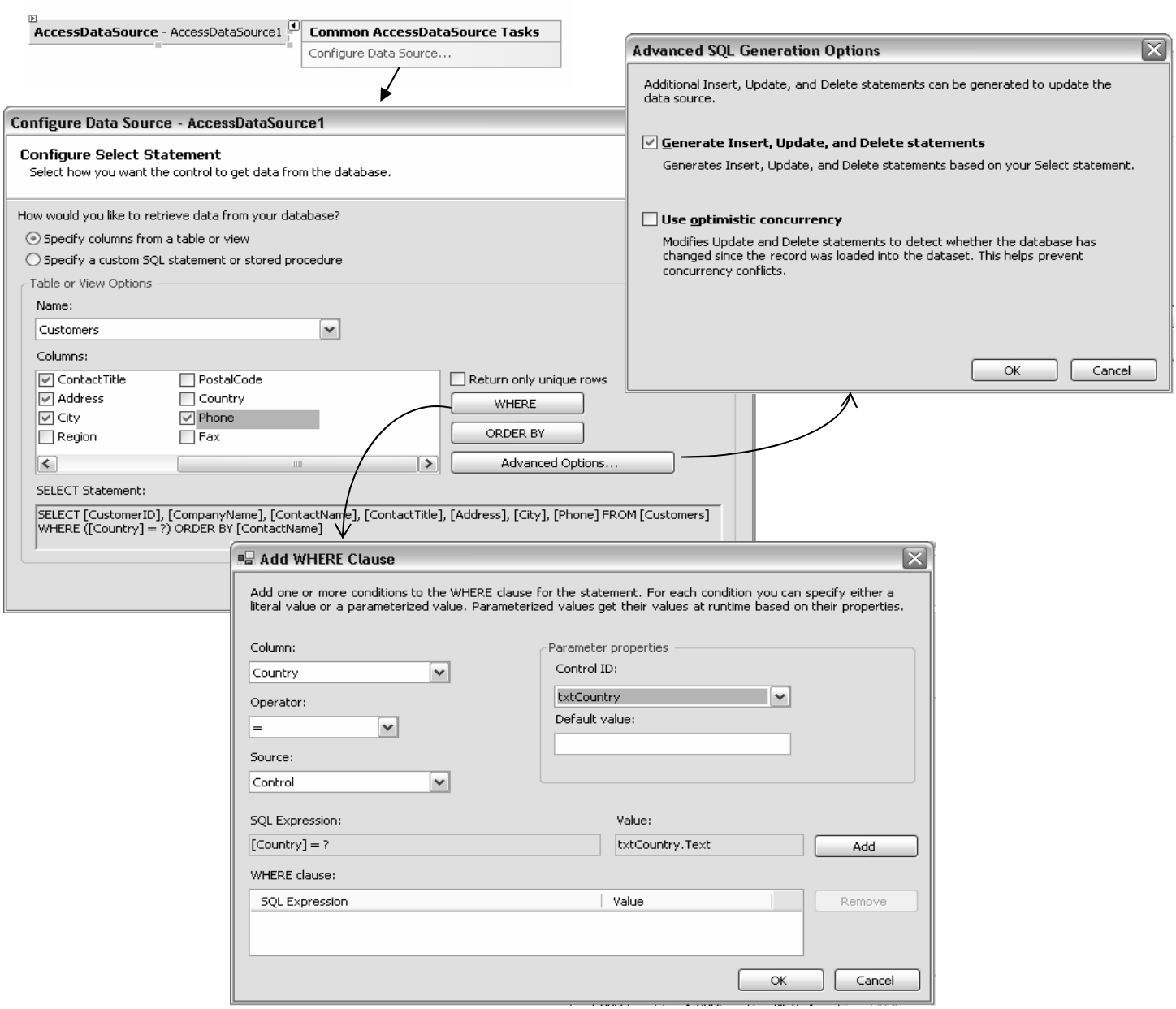

Figure 2. Configuration of an AccessDataSource.

SUMMARY

Visual Studio 2005 includes many new features not listed in this paper. The bottom line for class projects is that all the changes in Visual Studio mean students can develop more sophisticated, better looking and more robust web applications in a shorter period of time. This then frees more time for all other aspects of the development life cycle from project management, requirements and risk analysis to testing and implementation, documentation and user acceptance testing. The ability to place more emphasis on these aspects of the development process may help students understand that IS careers are much more than just coding and web page development and that

they are about finding solutions to business problems and helping organizations meet their strategic goals.

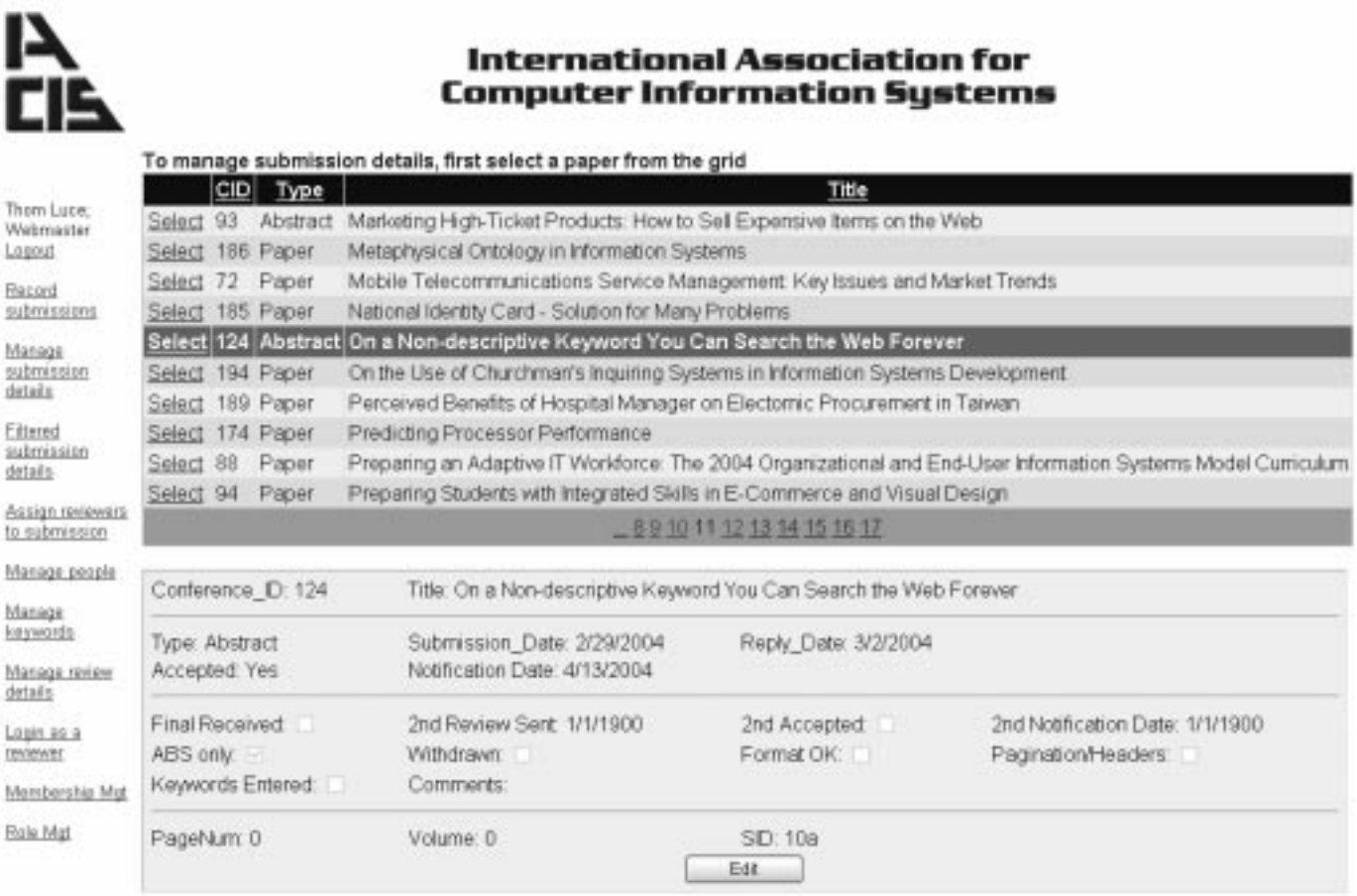

Figure 3. Web page showing a GridView (top) and a FormView

REFERENCES

1. Luce, T. (2001). *Developing Web Applications with Active Server Pages*, El Granda, CA: Scott/Jones, Inc.
2. MacDonald, M. (2002). *ASP.NET: The Complete Reference*, New York: McGraw-Hill/Osborne
3. Microsoft. (nd). Accessing Data with ADO.NET, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconAccessingDataWithADONET.asp, verified March 1, 2005.
4. Microsoft. (nd). ASP.NET 2005. http://www.asp.net/whidbey/" verified March 1, 2005.
5. Microsoft. (nd).Visual Studio 2005 Developer Center, http://lab.msdn.microsoft.com/vs2005/" verified March 1, 2005.
6. Microsoft. Preview of Web Development with Visual Studio 2005, http://www.asp.net/whidbey/whitepapers/VSWhidbeyOverview.aspx?tabindex=0&tabid=1 verified March 1, 2005.