Rhode Island College
Digital Commons @ RIC

Honors Projects Overview | Honors Projects

2006

Rhode Island College Mobile Course Catalog

Lianne Elsner
Rhode Island College, LianneElsner@gmail.com

Follow this and additional works at: http://digitalcommons.ric.edu/honors_projects

 Part of the Programming Languages and Compilers Commons

Recommended Citation
Elsner, Lianne, "Rhode Island College Mobile Course Catalog" (2006). *Honors Projects Overview*. 3.
http://digitalcommons.ric.edu/honors_projects/3

This Honors is brought to you for free and open access by the Honors Projects at Digital Commons @ RIC. It has been accepted for inclusion in Honors Projects Overview by an authorized administrator of Digital Commons @ RIC. For more information, please contact kayton@ric.edu.

Rhode Island College Mobile Course Catalog

Lianne Elsner
Computer Science Honors Project
Dr. Edward McDowell (Advisor)
April 2006

Table of Contents

Introduction

For my senior Honors Project in Computer Science, I have written a software package. The bulk of the package consists of a database application to run on Pocket PCs running Windows CE and Windows Mobile 2003/2003SE/5. Specifically, this program is an abridged version of the course catalog. Due to time restrictions, I could not create the entire catalog. Instead it includes the requirements for General Education and the major requirements for Computer Science (both BA and BS) and Mathematics (BA). It has been implemented using a "tree view." A tree view is a special type of window that allows the user to expand and collapse subcategories. For example, the "Computer Science BA" category has several subcategories, such as "Core Courses – All Required", "Advanced Programming – Choose One", "Programming Languages – Choose Two", and so on. Each of these subcategories contains a list of courses. When the user clicks on "Computer Science BA", a list of its subcategories appears. When a subcategory is clicked, the courses that fall under it appear. When a course is clicked, information about the course appears at the bottom of the screen. This information includes prerequisites, number of credits, and the catalog description.

The primary users of this software would be students at Rhode Island College. It allows them to have the course catalog for their year of admission in their pocket. They can instantly look up the requirements for their program and the necessary prerequisites. I decided to make this software because I know how tedious it is to find a catalog, find your major, then search through the back of the book for more information on a specific course. It is equally tedious to search for this information on RIConnect (especially for students with dial-up connections!) Finally, I have chosen the Pocket PC platform

because of its mobility. Pocket PCs are only slightly larger than a cell phone, and easy to carry around the campus. In addition, more and more people own Pocket PCs each year. I'm confidant that students will find this software very useful.

The second part of my software package is a maintenance program. This program consists of a main form that allows College staff to edit the database without needing to understand its design. The program allows staff members to add and delete majors, categories, and courses. It also allows them to edit the information of existing courses. I have designed this separate program for the Windows XP platform. Its purpose is to enable my software to retain its value as the years go on. My vision is that a staff member can download the database off of the RIC website, make the necessary changes, and then upload it back onto the site for students to download onto their Pocket PCs. Replacing the old database with the new one will update their software with the revised catalog.

The third, and possibly most important, piece of my package is a Microsoft Access database containing four linked tables of information. The first table contains majors. The second contains sub-categories ('Core Courses – All Required', 'Calculus Cognate – Choose One', and so on). The third contains the course names and descriptions. The final table contains the title for the Pocket PC tree (for example, "2005 – 2007 Catalog"). This database feeds both the catalog software and the maintenance software.

The final piece of my package is the documentation file for end users (see Appendix E.) This documentation includes information on how to use both pieces of software. It also includes installation instructions for the software components.

In addition to the previously described components, my software package utilizes the Microsoft eMbedded Visual Basic runtimes. These runtimes are required to run the catalog software on the Windows Mobile 2003/2003SE/5 platforms and to connect to the database of course information. This utility is available for free download from the Microsoft website. I have also included the runtimes on my program CD, in accordance with the rules set forth in the End User License Agreement (EULA) provided by Microsoft.

If this software suite were put into mainstream use, there would be two different versions of the software package. The first would be an "end-user" package on a CD to give to students, which would only include the catalog program, the database, the eMbedded Visual Basic runtimes, and a read-me file that describes only the catalog software. The second package, which I call the “administrative” package, would include the Administrative Tools program and read-me file in addition to the files of the “end-user” version. This is the package that would be given to the staff members responsible for updating the database. For the purpose of this project, I have created the “administrative” package. If this software were to be developed for use, both CDs would be put into production.

Related Works

In recent years, pocket personal computer (Pocket PC) technology has become increasingly commonplace, especially in professional and educational settings. College students especially are beginning to realize the usefulness of these powerful devices. A Pocket PC can be used to take notes, record lectures, and create documents to be beamed[1] for printing. A recent study done at Brigham Young University clearly shows the everyday usefulness of these devices. The study involved 26 secondary education students. Of these 26 students, only one previously owned a Pocket PC. Each student was issued a device, and asked to use it as he or she saw fit during the course of the semester. Once the students became familiar with the devices, they used them in a range of activities: managing their schedules, taking notes, reading E-books, and printing Word documents (Wangemann).

With the large number of uses for Pocket PCs, it's no surprise that so many college students are investing in such devices. There is, however, more functionality to these devices than simple data entry. Pocket PCs are also useful for retrieving data from the Internet. However, due to their small screen size, scaled down browser, and the lack of wireless Internet connections in many areas, Internet data retrieval is often a problem. For example, here at Rhode Island College, due to the limited range of the wireless network on campus and the nature of the RIConnect website[2], it is impossible for a RIC student to access course catalog information using a Pocket PC. Using the technology

[1] Beam – to send data via an Infrared connection from a Pocket PC to an Infrared-enabled desktop computer or printer

[2] Due to the programming of the website, it cannot be displayed using Pocket Internet Explorer. This browser is optimized for standard HTML websites only.

that is currently available to me[3] I have attempted to solve this problem through the use of offline data storage.

I am certainly not the first to realize this flaw in Pocket PC technology. Doron Cohen, along with five fellow IBM researchers, proposed a solution for this problem in 2002. Their solution involved storing information from the Internet directly into the device's memory to be accessed offline (Cohen, et. al., 627). The information would then be updated through user-initiated synchronization with a desktop computer. The result is fast access to current data without a constant Internet connection (Cohen, et. al., 628). This is the general idea behind the approach that I have taken in solving the problem at hand.

This same approach has been taken by AvantGo, a successful piece of software that allows Pocket PC users to download web content from the Internet onto their devices. News articles, maps, movie times, and other user-selected content are stored in the device's memory and remain there until the user elects to update it (AvantGo). Currently, 7 million Pocket PC owners use this software (Cha). A major advantage of software of this nature is that you are "able to read [information] straight through and not have to scroll left and right to get the whole page" on the small Pocket PC screen (Cha). This software allows the power of the Internet to be used at the Pocket PC owner's convenience.

[3] Computer technology changes rapidly. Therefore I will be using the technology at my disposal as of the time my proposal was submitted (April 15th, 2005).

Design of the Solution

Based on the needs of the students who will be using my mobile course catalog, I chose to write the Pocket PC software using eMbedded Visual Basic (eVB). According to Mark Dixon, this language is the most practical for Pocket PC development (Dixon, 272). He bases this claim on Visual Basic's reputation as a powerful language for user-interface programming (p. 283). My second, and most important, reason for choosing this particular language lies in the nature of my project. Unfortunately, the .NET suite of languages does not support ADOCE, which is required to connect to a database local to the device[4]. Instead, these languages require an SQL Server database stored on a remote server[5] (Microsoft Corporation). As I mentioned earlier, a remote database connection would defeat the purpose of my project, because a constant Internet connection would be required to use the software.

I spent many long hours during November of 2004 working to solve this problem. I read online articles and forums, downloaded and tested programming software, and even attempted to reformat my project idea. During my research, I discovered that eMbedded Visual Basic provides support for local database connectivity using ADOCE. In the end, I decided that eMbedded Visual Basic was the best language choice based on the needs of my project.

In order to make this software useful to as many people as possible, it needs to be easily updatable. Maintainability is a major consideration of software engineering (Stiller and LeBlanc, 304), and was therefore considered in my project. My eMbedded Visual

[4] "Local to the device" – stored in the device's memory, as opposed to on the Internet or a network
[5] "Stored on a remote server" – stored on *another* computer that the device must connect to via the Internet or a network

Basic code has been well documented, thus allowing future programmers to port the code[6] to a .NET language if the resources become available. In addition, I have created a maintenance program for the database, which runs on a desktop computer. This program allows college staff to easily update the database, which will make my software retain its value as changes are made to the course catalog.

Another software engineering consideration, usability, was important in the design of the maintenance program. Specifically, usability refers to the ease with which the end-user can use the software (Stiller and LeBlanc, 304). To increase usability, I have created the program in such a way that the end-user does not need to know anything about the database in order to edit it.

I have written this portion of the program in Visual Basic.NET, which will run on all computers running the Windows XP operating system. Faculty and staff at Rhode Island College have easy access to this technology. Therefore, I foresee no future compatibility issues with this portion of the software package.

[6] "Port the code" – translate

Details of the Solution

As stated in the previous section, I designed my suite of programs with optimal usability in mind, designing the user interfaces in such a way that the programs would be simple to use. I also created the layouts with the screen space of the target device in mind.

For the Pocket PC program (see Figure 1, Appendices A and B), I constructed the interface out of several objects. The main listing of majors, subcategories, and courses uses a tree view object, which allowed me to create a standard tree listing of the database tables. Upon loading, the program connects to the database and reads the four tables into record set objects. A series of loops then move the data from the record sets to the correct tree nodes. Each node is identified by a key which is stored as part of the node object, and is required for the subroutine that looks up the corresponding data for each course. The tree view object handles the processes of expanding and collapsing the nodes behind the scenes.

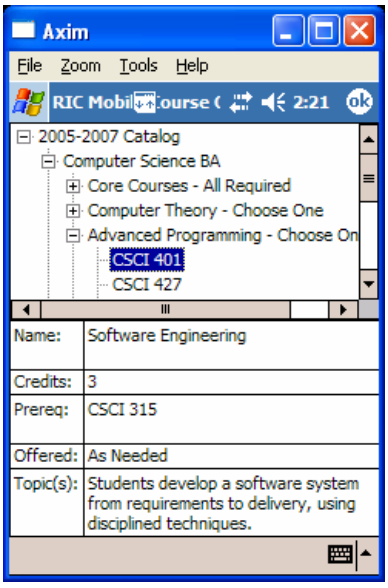

Figure 1: The Pocket PC Interface

The second half of the Pocket PC screen consists of a series of text box objects that display the data for the selected course, and that are populated from the database. This is taken care of by a special subroutine that I have created that handles the event fired by the tree view object when an item in the tree is selected. If the user clicks on the name of a course, the program searches for the matching item in the Courses table of the database using the record set object. Once the data is found, each column of the table is placed into the corresponding text box. If the user clicks on the root node, a major, or a subcategory, my subroutine ignores the action and does nothing.

The interface for the desktop program (see Figure 2, Appendices C and D) is significantly more complex. It consists of a main form which houses several frame objects. Each frame object represents a table from the database, and contains all of the input and output objects needed to perform actions on that table. The frames are swapped in and out depending on the button that the user presses, which was done to make the interface look clean and to reduce on-screen clutter. The click event handler for each button decides which frames should be visible and which should be hidden.

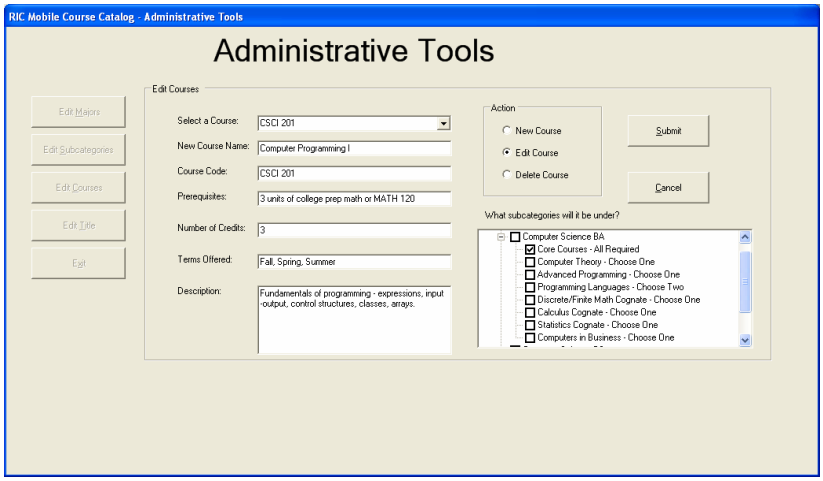

Figure 2: The Desktop Interface

Upon loading of the main form, an OLE database connection object is created along with four OLE data adapters, which are used to connect to the individual tables. The data from each table is then read into a data table. The connection object is opened upon loading, and closed when the form's closing event is fired.

When each button is clicked, the corresponding frame is made visible, and the dropdown boxes are populated from the database. An array keeps track of the primary key for each item in the dropdown box, so that when one item is clicked the corresponding data can be looked up in the data table. The data is presented to the user in a series of text boxes. If the frame is in delete mode, the text boxes are read only. If it is in edit mode, the user can now make the necessary changes to the data.

Once the user has made the necessary changes, the new data is saved to the database. This is done when the "Submit" button corresponding to the active frame is clicked. When these subroutines are called, the program looks at the radio buttons on the active frame to determine which operation to perform on the data (e.g., Add, Edit, or Delete). Once that has been decided, the program validates the input. In this program, validation is mainly concerned with verifying that all required fields contain data. Individual text boxes have their max length property set to the maximum number of characters the database field (and corresponding Pocket PC textbox) can handle.

Next, the program performs the specified operation(s) on the database. In all cases, the user is notified of a success or failure. The user is then asked if he/she wants to continue working with the active frame. If 'Yes' is selected, the form is reset and made ready for fresh user input. If 'No' is selected, the form is reset and the user is taken back to the main menu screen.

User Testing

Another of my major concerns in designing this program was compatibility. In order for my program to be useful, it must work on as many systems as possible. This was more of a concern with my Pocket PC program than with my desktop program. Because I only have access to a single Pocket PC, I decided to turn to the web for feedback on the compatibility of my program (see Appendix F). I posted on a popular Pocket PC forum, Aximsite.com, and asked for volunteers to Beta test my program (Elsner). I received several responses from users of a variety of Pocket PC devices.

I am pleased to report that my program worked on all of the Pocket PCs tested, including my own. Of the group of testers there was one device that I did not expect my program to work on. This device was a PPC-6700 model Pocket PC Phone Edition (made by Sprint) running Windows Mobile 5. According to Microsoft, eMbedded Visual Basic programs are not compatible with Windows Mobile 5. According to James Pratt, a Product Manager for Microsoft, "the important thing to realize is that [the eVB] runtime will not work on the next release of Windows Mobile software so you definitely shouldn't create any new apps using eVB" (Pratt). One of my Beta testers, a man named Max, disproved this statement. When I asked him if he had any problems running my program, he said, "no." (See Appendix F, Eval. Form 1.)

When testing my desktop program, I performed the Beta testing myself. I tested it on three different computers, all running Windows XP (SP1 or SP2), each having different hardware configurations. The program installed and ran flawlessly on each machine. There are no documented compatibility issues between Visual Basic.NET code and Windows XP computers, so I saw no reason to pursue the testing further.

Conclusions and Future Work

The end result of this project is a fully functional software package that allows end users to easily view or update the Rhode Island College Mobile Course Catalog. As promised, the programs are simple to use. Staff members can update the database using my interface. No knowledge of the database design is required. Students and faculty can quickly and easily look up information for a specific course without having to search through PeopleSoft or a large paper catalog.

Although the software works exactly as promised, there is certainly room for improvement. If given more time, I would have improved upon the way that the software interacts with the database. Specifically, I would change the code so that the database is completely hidden from the user. In the Administrative Tools program, once the user is done editing the database, there could be a button on the screen that allows the user to export the database. The program would then create an exact copy of the internal ("hidden") database, convert it to a .cdb file and save it on the user's desktop. This would prevent the potential problem of a user opening the database with Access and attempting to manually edit the data. Without knowing the design of the database, the user could corrupt the data and make it unreadable by my program.

For the Pocket PC program, I would add a menu item that allows the user to import a database. This would allow the user to select any database on his/her own device to import into the RIC Mobile Catalog. The program would take the selected database and use it as a replacement for its internal database. Error checking within the program would verify that the table structure of the database matches my design. Once again, this would remove the potential problems that could result from a user tampering

with the database file (e.g., renaming or moving it, or opening it with a third party Pocket PC database program). I would also add functionality that would allow the user to store multiple course databases on the device (i.e., catalogs from different years), and select the one to view via a menu. This would increase the usability of the program, especially for faculty members who are advising students who began their studies in different years.

Since I will not be able to make these improvements to the software at this time, I carefully commented my code so that future programmers may interpret and improve upon it down the line. To help make it feasibly for the College to implement this software, I have very thoroughly commented the code. Such high quality documentation is crucial in today's rapidly changing technology.

Acknowledgements

I would like to thank Dr. Edward McDowell for his support and guidance as I worked on this project. I would also like to thank the rest of the Computer Science faculty, especially Dr. Ying Zhou, Dr. Kathryn Sanders, and Dr. Roger Simons for their guidance over the past four years. Finally, I would like to thank Thomas Quinn for his editorial advice on my thesis.

Works Cited

AvantGo. <http://www.avantgo.com>.

Cha, Bonnie. "AvantGo 2005." CNET Editor's Take. <http://reviews.cnet.com/AvantGo_2005/4505-3638_7-31272316-2.html.> 1 February 2005.

Cohen, Doron, et. al. "Personalized Pocket Directories for Mobile Devices." WWW2002. Honolulu, HI: May 2002. 627-628.

Dixon, Mark R. "Creating a Portable Data-Collection System with Microsoft eMbedded Visual Tools for the Pocket PC." Journal of Applied Behavior Analysis 36.2 (Summer 2003): 271-284.

Elsner, Lianne. "Beta Testers Wanted!" 26 September 2005. Online posting. Aximsite. 4 March 2006. < http://www.aximsite.com/boards/showthread.php?t=99319.>

Microsoft Corporation. End User License Agreement for Microsoft Software: Software File = msvbppc.armv4.cab. <http://msdn.microsoft.com/mobility/windowsmobile/downloads/evb_eula.aspx.>

Microsoft Corporation. What's New for Developers in Windows Mobile 2003 Software for Pocket PCs. <http://msdn.microsoft.com/library/en-us/dnppc2k3/html/winmob03.asp>: June 2003.

Pratt, James. "re: The eVB download page." 25 January 2005. Online posting. Windows Mobile Team Blog. 4 March 2006. < http://blogs.msdn.com/windowsmobile/archive/2005/01/03/345995.aspx#360419.>

Stiller, Evelyn and Cathie LeBlanc. Project-Based Software Engineering: An Object Oriented Approach. Boston, MA: Addison-Wesley, 2002.

Wangemann PH.D, Paul, Nina Lewis, and David A. Squires. "Portable Technology Comes of Age: The Utilization of Handhelds in a Pilot Teacher Education Program." T. H. E. Journal 31.4 (Nov 2003): 26-30.

Appendix A:

Pocket PC Screenshots

Appendix B:

Pocket PC Source Code

Appendix C:

Desktop Screenshots

Appendix D:

Desktop Source Code

Appendix E:

“Read-me” Document for Software Suite

Appendix F:

Pocket PC Beta Testing Results