ABSTRACT

The very nature of software is that it very quickly outgrows the initial specifications and goals for which it was built. Good, upgradeable software is perhaps the most ideal example of how gracefully mathematics and formal logic can be combined in order to explore any problem domain. The most attractive facet of any piece of software is its adaptability. Every organization needs its software to keep up with its changing paradigm and demand. This application is aimed at helping those companies create an entire data entry application in precisely this way: easy, fast and customized to their needs. The application will create a framework that recognizes the user settings and configuration, and then builds the user interface screen with the appropriate controls and functionality. This system could be used by organizations requiring a data entry application with direct, interactive user input and data validation. The goal is to develop a system that will maintain its scope and functionality across a wide range of applicability.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. BACKGROUND AND RATIONALE

1.1 Current Application and its Drawbacks

Traditionally, when a private business, government organization, hospital or a financial institution needs software, they look for a system that solves their present situation or problem. But when they later find that their needs have outgrown the software that they bought, they are obliged to return to the original vendor in order to purchase typically expensive upgrades. Customizing this upgrade costs even more and it is a lengthy task that involves massive inconveniences with respect to time and money.

In the market today, several software vendors cater to organizations with varied requirements. When a software vendor is assigned a task, he/she must start from scratch and design the screen as per the user's specification. This may take a long time for the vendor to complete the task. The functionality of an application may change many times during its life cycle, due to maintenance and enhancement. For even a minor change on the screen layout, the software vendor has to manage manpower that results in a loss of time and money for both parties.

As computers become more accessible, the problem of designing dynamic interfaces becomes more severe. Many new users expect interacting with their application to be as natural and intuitive as interacting with a person, but current applications such as Microsoft Visual Studio are artificial and constraining [Myers 2003]. One particular weakness with the application is its static nature. The application is developed to interact identically with all users and for a range of tasks, without considering differences in knowledge, preferences, and purpose. An application should be constructed once and be (dynamically) configured to suit the environment in which it executes. Tools such as

Microsoft Visual Studio can be used for constructing configurable systems which meet the user requirements but there is a need for a completely flexible and portable system for the user [Little 1998]. The application does not provide a dynamic and configurable interface for the user. The framework on which the application is built does not allow it to be portable and executed across all Windows operating systems.

Commercially available products such as Visual Interdev provide a framework and a set of components to design screens for the user [Balena 2002]. The drawback is that it doesn't provide user flexibility. In other words, the user is totally dependant on the software developer for any changes. Although there are many custom data entry applications available, none of them offers a fully configurable interface like the one this project offers.

1.2 Advantages of the new system

This application utilizes an XML (Extensible Markup Language) file to provide a simple Graphical User Interface (GUI) for choosing screen layout based on user needs. In this way, the user is given full control over the look and feel of the application [Francis 2002]. After saving the settings, the field parameters are written to an XML file in the application directory.

When the application is invoked, the framework reads the XML file and reconstructs itself from it. After a period of time, if the user wishes to change a field or move the field around, they do not have to depend on the software vendor to achieve this simple task. This saves a lot of time and money in maintenance and enhancement. It can help a software programmer by making his job easier and more effective. The system can be used to serve other departments or needs with very little modification. It can be used

for different clients and business with very few changes. This is a time saving and cost effective solution for many organizations.

Traditionally, applications are developed based on design layouts and the requirements. In contrast, this project transfers control to the user, who would place desired fields on the screen. The application, however, allows the user to design the user interface, thereby negating chances of misinterpretation. All the built-in functionality is optional and can be turned on/ off in the configuration file by the user. This saves a lot of time for the developer, since the screen is already designed as per user specifications and hence, the time needed to add a new feature is far less than starting the whole process from the beginning [Ellmer 1996].

2. NARRATIVE

2.1 Overview

The application has been developed using VB.NET, XML and SQL Server. It is a Windows based application that can be installed and executed from a centralized location.

2.2 Features

The scope of this system (**XML Generator**) is very broad. It can be used by virtually every business that needs to perform data entry. It accepts entries from the user, validates them, performs calculations if needed and then saves the result in a database of a user's choice. The user can enter the data in a text format, select items from drop down box or combo box, numeric spin box, and date selection control, etc. It also accepts scanned images to be saved along with the particular user's record. Such a system can be very attractive in school districts, jails, government agencies, and many other organizations where it is very important to capture detailed personal information and other documents related to a person.

Security was a major feature to evaluate for this system. The long term and effective success was dependent on how the security issues are handled by the system [Ambler 1995]. Since the application was also developed to be used by Government Agencies, the access information needs to be secure and encrypted. Good encryption and decryption techniques were used to protect the database connection string, the user name, password and the important configuration information's were stored in secure places such as in Windows Registry

Secure Hashing Algorithm (SHA1) and Message Digest Algorithm (MD5) were used to encrypt and decrypt any secure text used in the application. SHA1 takes a very large message and produces a 160-bit message digest. The MD5 algorithm takes as input a message of arbitrary length and produces as output a 128-bit "message digest" of the input. It is not a good practice to put them clear in database since storing passwords as a clear text could cause potential breach to the application For example, it is now shown how these algorithms were implemented to secure passwords in database. Consider a table with two columns, "Name" and "Password". Assume the passwords stored in the table are in clear text format. We will now apply hashing algorithms to produce message digest and secure them. .Net provides various namespaces to implement SHA1 and MD5 Hash Algorithms. **System.Security.Cryptography** namespace can be used for producing SHA1 and MD5 message digests. A clear text can be encrypted to SHA1 and MD5 correspondingly by simple calling the statement like

```
FormsAuthentication.HashPasswordForStoringInConfigFile(Me.txtPassword.Text.Trim(), "SHA1")
FormsAuthentication.HashPasswordForStoringInConfigFile(Me.txtPassword.Text.Trim()," MD5")
```

The application was tested with various possibilities like blank passwords, passwords with special characters, long and short passwords. The texts were encrypted and decrypted using the same algorithm to make sure it produced exact result. The application passed all the security tests implemented for the passwords and other sensitive information.

2.3 Description of User Interface

The following sections describe the controls and their features that are available for the user for building the screen. The user should decide the controls that best suit their needs for the generated application. It is assumed throughout this document that the user that is designing the screens using **XML Generator** application will be one using the generated application.

2.3.1 Creating a new layout

When the **XML Generator** is first invoked, the main screen will appear as shown in Figure 2.1. The Tool box on the right is populated with the list of Components or Controls that can be used and a form control is displayed at the center of the main screen. Other controls that are supported by **XML Generator** are Buttons, Labels, Picture Box, Tab Control, Text Box and Reoccurrence. The User is allowed to drag and drop the controls from the right to the form control. The default properties for each control are automatically loaded at run time. They can be viewed and modified in the properties window shown at the bottom right corner of the main screen.

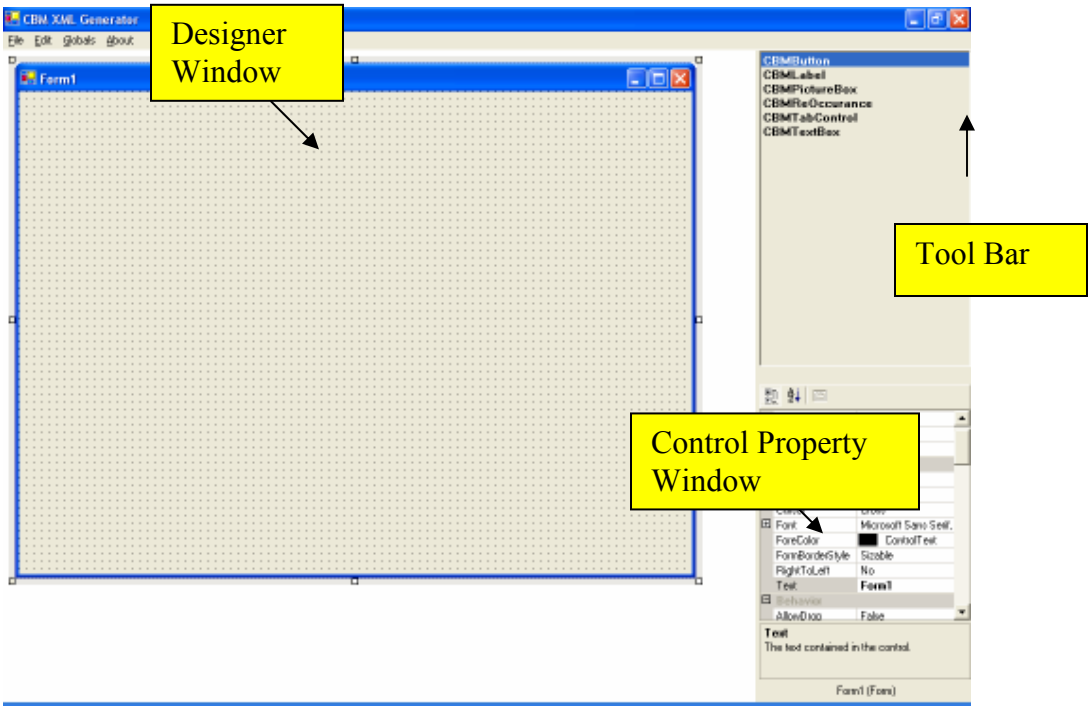

Figure 2.1: Main Screen.

2.3.1.1 Form Control

As shown in Figure 2.2, a blank designer window can be created by clicking on **New** button from the Menu Bar on the top. The form can be resized to any size by clicking and dragging on the window resize points as shown in Figure 2.3

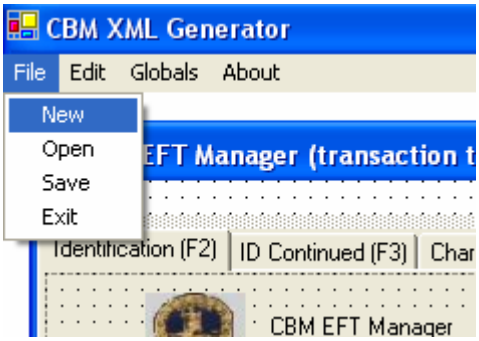

Figure 2.2 New Menu item.

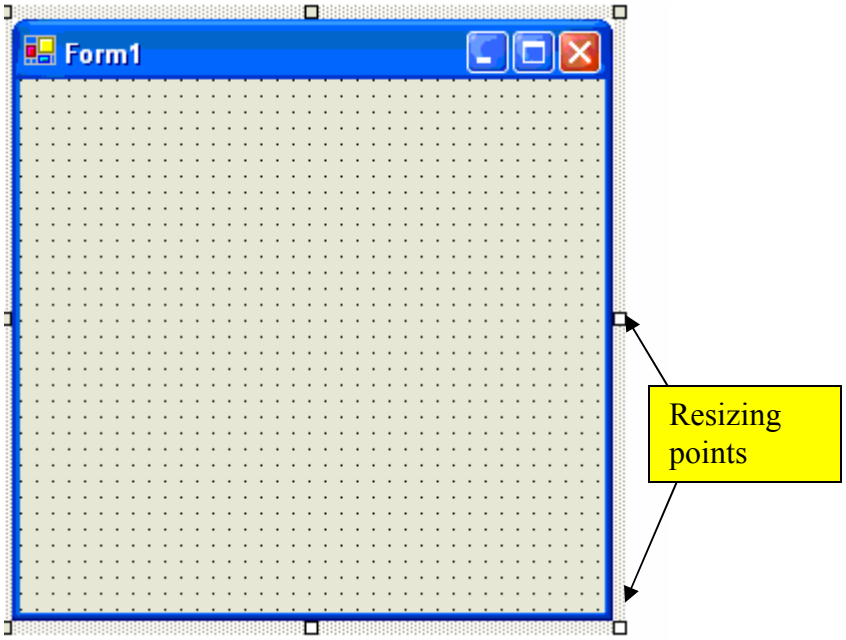

Figure 2.3 Form Resizing points.

The default properties for the form control can be viewed on the properties window. Some of the properties that can be set for the form control are

Back Color - The background color of the form can be changed by choosing the desired color from the color palette.

Background Image- An image can be selected to be the background of the form by selecting an image of the known formats (JPEG, GIF or BMP).

Height and Width- The form can be resized to a desired size either by dragging on the window resize points or by manually entering the measurement in Pixels.

Name- The name of the control.

Tab Index- It is the order in which the tabs in a form should go as they navigate through the form. If a tab index of a text box is set to 1 then the application will tab to this text box first. The tabindex should be unique, i.e., there should be any other control in the form with the same number.

Text- The text that appears on the top of the window can be set in this property.

Icon- The  icon that appears on the window can be changed by browsing and selecting an icon from disk.

Start Position- The default position of the form when the application is invoked can be set here. The property can be set to display at center of the screen or can be manually set by the user.

Font- The font of any text that would be set on the form can be set here.

Form Border Style- The border for the form can be set here. The border can be fixed single line, fixed 3D or none.

2.3.1.2 Tab Control

Most application will have multiple screens to display different categories of a data entry screen. Here Tab controls are provided to make the task easier by allowing multiple layouts in one control. A tab control can be selected from the Tool box and dragged onto the form. After adding, the form will be as shown on Figure 2.4.

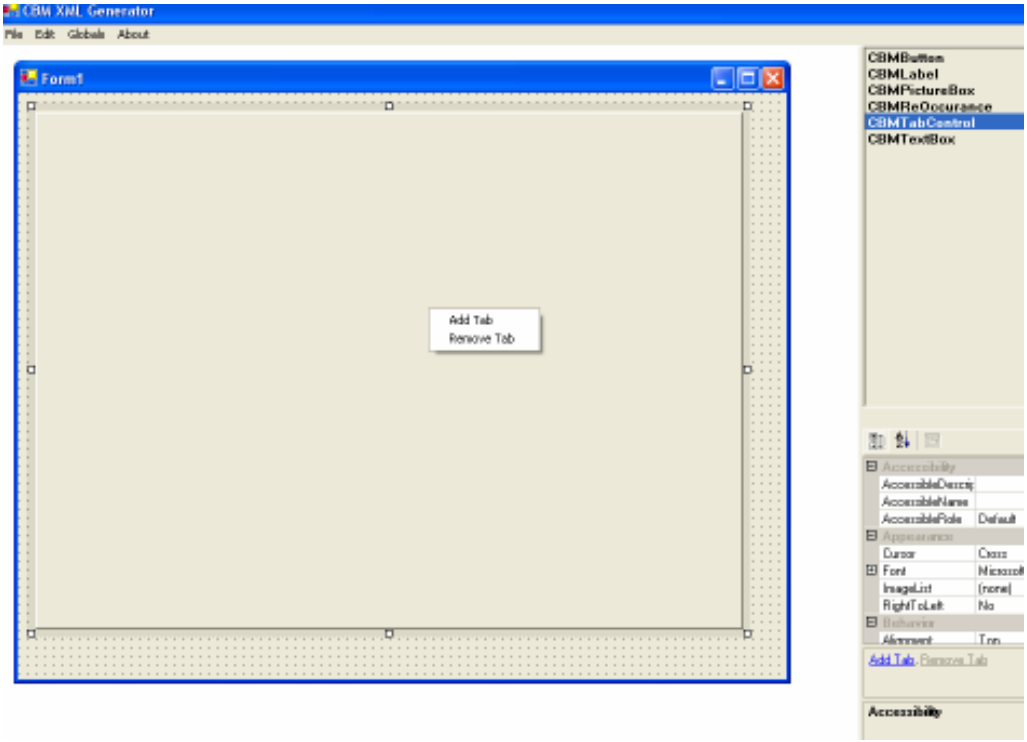

Figure 2.4 Tab control on a form.

Multiple Tab screens can be added by right clicking on the form and selecting **Add Tab** from the two options provided on the menu as shown in Figure 2.4. Each tab page can be assumed as separate screens in an application. The tab pages will appear as shown in Figure 2.5. Similarly any tab page can be removed from the form by selecting the **Remove Tab** option.

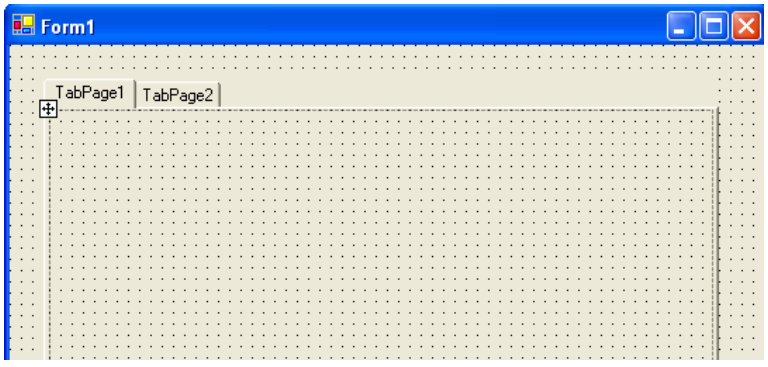

Figure 2.5 Tab pages.

Some of the properties that can be set for a tab controls are listed as follows:

Tab Pages- The tab pages can be added and all the properties for the tab pages can be set in this property. All the properties of the tab pages are the same as that of the form.

Size- The height and width for the tab controls can be set here.

Location- The location of the tab control on the screen can also be set.

Tab Index- Indicates tab index of the tab control on the form.

Font- Specifies the font to set on all the text in the tab control. This will override the font property set for the form control.

Anchor- This property sets the distance that the tab control should be from the corner of the form when the control is resized.

2.3.1.3 Picture Box

An image can be added to a form or a tab page by using a Picture Box. Picture box control can be only a tab or a form control. The images that can be loaded into this control can be of any one of the known image format (JPEG, BMP or GIF). The images can only be chosen from disk and not copy-pasted directly into the control. After adding the picture box to the tab page, the screen will appear as shown in Figure 2.6. It can be resized by dragging on the resize points for the control. The user should make sure that the image will be able to fit the picture box dimension without any loss of quality. If the user wishes that the control could be set to automatically resize depending on the image, the control will be automatically resized to best fit the image.

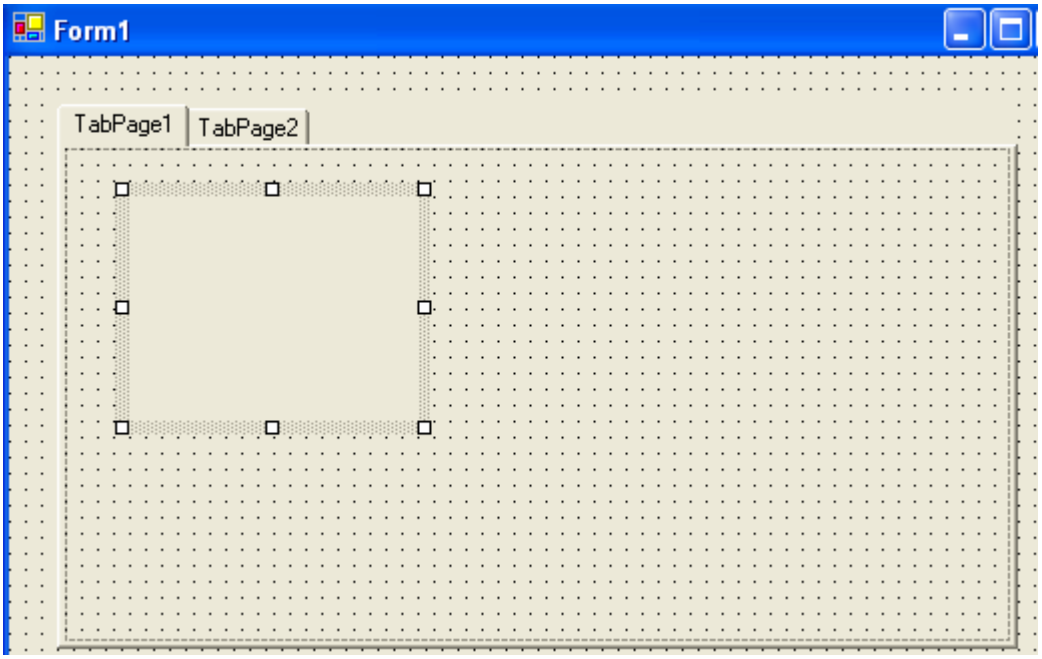

Figure 2.6 Picture box before the image was loaded.

Some of the properties that can be set for the picture box control are:

Image- The Image that should appear in the picture box control can be set by clicking on the image property as shown in Figure 2.7. A window will appear prompting the user to choose the image from disk as shown below. Then the selected images will be displayed as shown in Figure 2.8.

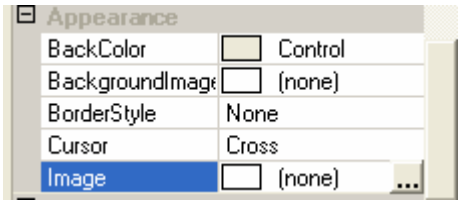

Figure 2.7 Property window for Picture box.

Figure 2.8 Picture box after the image was loaded from disk.

Size Mode- Some of the options for the size mode are Normal, Auto Size, Stretch Image, and Center Image.

a) Normal- The image will show up in the control inside the control without any modification.

b) Auto Size- The control will automatically resize the control to fit the image.

c) Stretch Image- The Image will stretch to fit the control.

d) Center Image- The image will be centered on the control.

Border Style- This property specifies the border that should be set for the control. If the border is set to fixed single then a thin border will appear around the control. If set to fixed 3D then the border will have a 3D appearance.

Size- The height and width for the picture box can be set here.

Location- The location of the picture box on the screen.

Tab Index- Indicates tab index of the picture box control on the form.

2.3.1.4 Label Control

Label control can be used to display text in a form. As shown in Figure 2.9, labels can be placed on a form by dragging the control from the Tool bar and placing it on the designer surface. Then the control can be resized to the desired ratio by clicking and dragging on the control resize points.

Some of the important properties of a label control are

Size- The height and width for the label control can be set here.

Location- The location of the label control on the screen.

Tab Index- Indicates tab index of the label control on the form.

Visible- It can be set to true if the label should be initially visible when the label control is placed on the form or false if they are to be invisible.

Name- Unique name of the label control on the form.

Text- This is the caption of the text that needs to appear on the control.

Back Color- This is used to specify the color that should be used for the control. The color can be chosen from the color palette.

Back Ground Image- An image can be chosen from disk to be the back ground of the label control. The control accepts images in JPEG, BMP and BMP file formats.

Font- Specifies the font to set for the label control. This will override the font property set for the parent control.

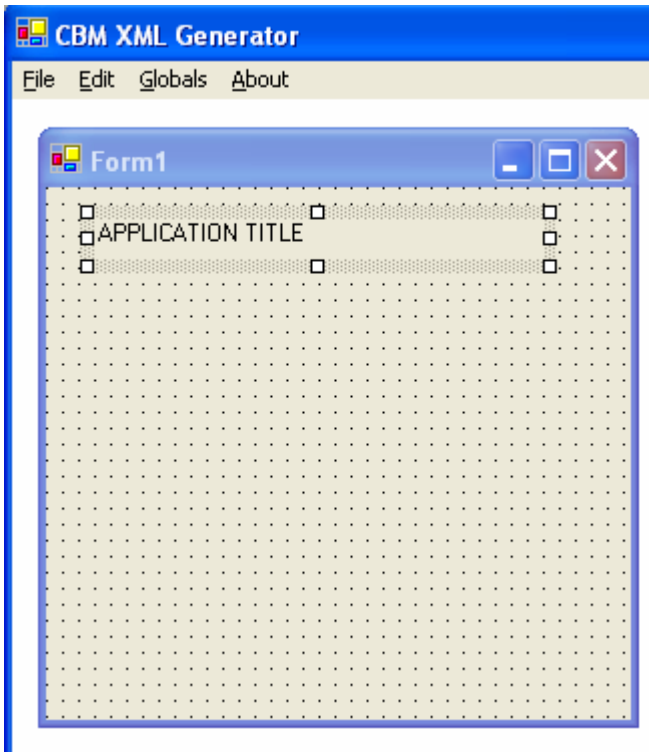

Figure 2.9 label control on a form.

2.3.1.5 Text Box Control

Text Box control can be used to accept inputs from the user. The control could accept a single or multiple lines of text from the user. This is a key control for all data entry applications. The control can be placed on the form by dragging the control from the tool bar and placing on the form. The controls can be resized by using the window resize points or by manually setting the dimensions. Some basic validations, such as maximum and minimum length of the text, could also be set on these controls.

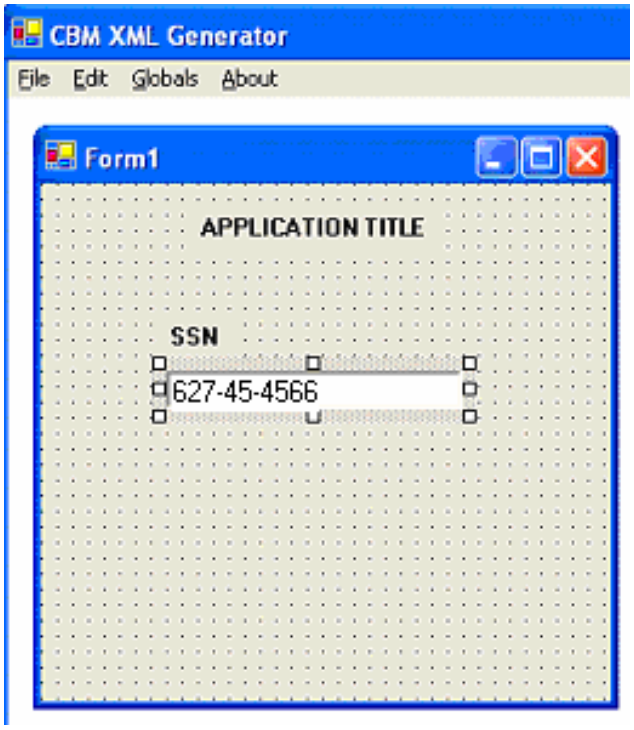

Figure 2.10 Text box control on a form.

The most commonly used properties for this control are shown below

Size- The height and width for the text box can be set here.

Location- The location of the text box on the screen.

Tab Index- Indicates tab index of the text box on the form.

Visible- It can be set to true if the label should be initially visible when the text box is placed on the form or false if they are to be invisible.

Name- It is a unique name for identifying the text box on the form.

Text- This is the caption of the text that needs to appear on the text box.

Back Color- This is used to specify the color that should be used for the control. The color can be chosen from the color palette.

Font- Specifies the font to set for the text box. This will override the font property set for the parent control.

Caps Lock- when the user leaves the control the text in the control is changed to upper case if the property is set to true or lower case if it is set to false.

Enter Key- If the property is set to true then enter key can be used to tab out of the fields. If false, the user can only use tab to leave the field.

Max Length- It specifies the maximum number if characters allowed in the text box. It can be very useful for restricting invalid entries made by the user. The control will not allow more characters to be keyed than the set limit.

Mult line- If the property is set to true then the text box can accept multiple lines of text.

Read Only- If the option is set to true then the text in the control can only be read but cannot be edited.

Required Field- If the option is set to true then the application will not let you leave the control without a valid entry.

Regex value- To make sure that the users enter a valid entry before exiting out of a text field, a property called Regex property was used. This denotes a regular expression that specified the valid combination allowed in a text box. For example, a field that accepts Social Security Number could accept hyphens or period separating the entry. In cases like those a regular expression could be formed that will accept only the required number of digits with the possible combinations. This is one of the powerful tools available in the XML generator application. Composing a regular expression requires intermediate or advanced knowledge of the subject, but a well formed regular expression eliminates the need for multiple line of validation check to ensure that the field has been keyed in

correctly. Only if the entry made by the user passes the regular expression validation, the users will be allowed to exit out of the field.

2.3.1.6 Button Control

This control can be used to invoke a method when it is clicked. It can be used to invoke an action that performs the desired function for the user. The control can be placed on the form by dragging the control from the Tool bar to the designer window. The dimensions of the control can be manually set in the property window or dragging the control to the desired ratio.

Some of the commonly used properties of the button control are:

Size- The height and width for the button can be set here.

Location- The location of the button on the screen.

Tab Index- Indicates tab index of the button on the form.

Visible- It can be set to true if the button should be initially visible when the button is placed on the form or false if they are to be invisible.

Name- It is a unique name for identifying the button on the form.

Text- This is the caption of the text that needs to appear on the button.

Back Color- This is used to specify the color that should be used for the control. The color can be chosen from the color palette.

Font- Specifies the font to set for the button. This will override the font property set for the parent control.

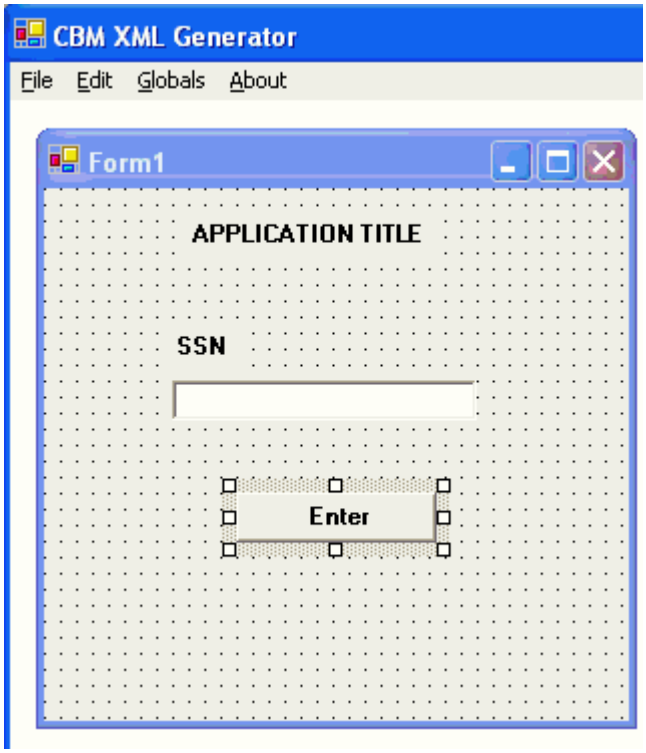

Figure 2.11 Button Control on a form.

The above screen shown in Figure 2.11 could be assumed to be a sample application that accepts the Social Security Number in the text box control. Once they click **Enter** button, they could perform a look up or an action based on the value entered in the **SSN** field.

2.3.1.7 Reoccurring Control

The control can be used to group repeated set of controls on a screen. If a set of controls are repeated many times in a screen for displaying data with the same structure then this control could be very useful. As shown in Figure 2.12, the application accepts the personal information of the passengers in a car. The controls are capturing the same set of data on the screen. Assuming they had over 20 passengers in the car then the

controls would have occupied the entire screen. The same case can be accomplished very easily and efficiently using Reoccurring control.

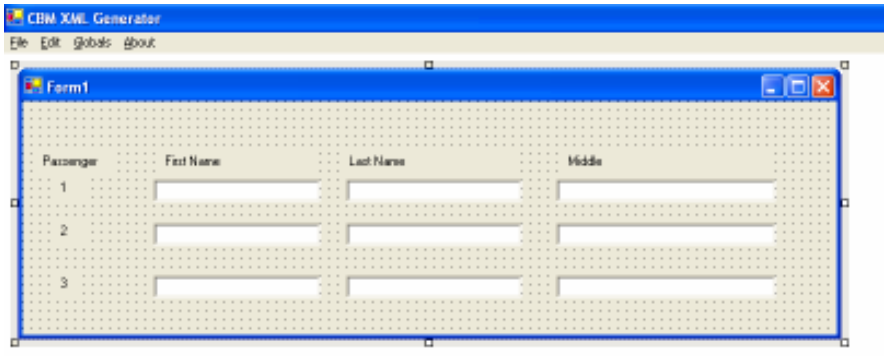

Figure 2.12 Repeating set of fields on a form.

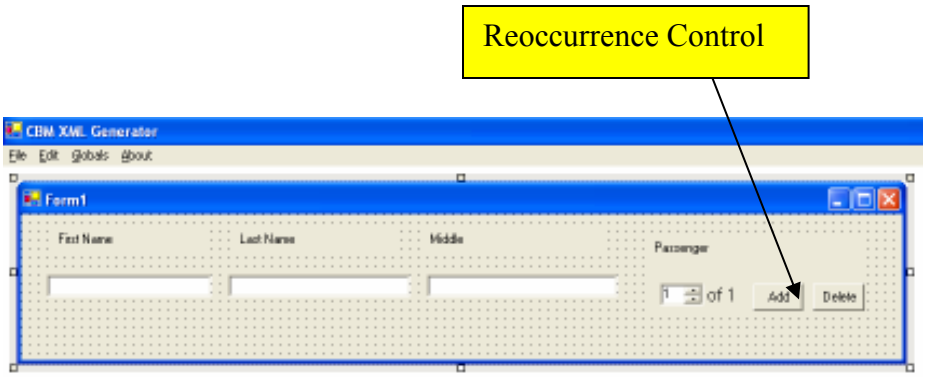

Figure 2.13 Reoccurrence control.

Figure 2.13 shows the same repeated set of controls grouped using Reoccurring control. There are no limitations on the number of repeated fields that could be grouped in this control. A new set of repeated fields could be created by clicking on the **Add** button shown in Figure 2.13. To remove particular instance of a set of fields, the appropriate fields could be scrolled to by clicking on the scroller as pointed in the figure and pressing on **Delete.**

Some of the properties that can be set for this control are:

Size- The height and width for the control can be set here.

Location- The location of the control on the screen.

Tab Index- Indicates tab index of the control on the form.

Visible- It can be set to true if the control should be initially visible when the button is placed on the form or false if they are to be invisible.

Name- It is a unique name for identifying the control on the form.

Reoccurring Text- This is the caption of the text that needs to appear on the reoccurring control.

Back Color- This is used to specify the color that should be used for the reoccurring control. The color can be chosen from the color palette.

Font- Specifies the font to set for the button. This will override the font property set for the parent control.

2.3.2 Saving the layout

The layout designed on the screen can be saved by clicking on the save menu on the Menu bar at the top. When the save menu is pressed the application loops through all the fields on the screen and make sure they satisfy the application requirements (a valid name should be assigned to every control in a form).

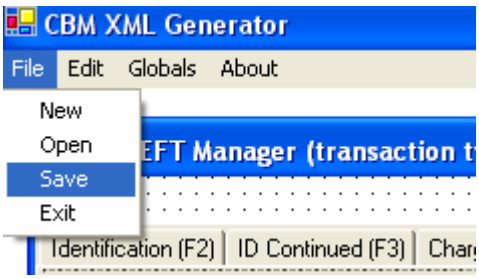

Figure 2.14 Save menu item.

The user is then prompted to choose the name of the XML file to save to. The user could save the XML file to the desired location.

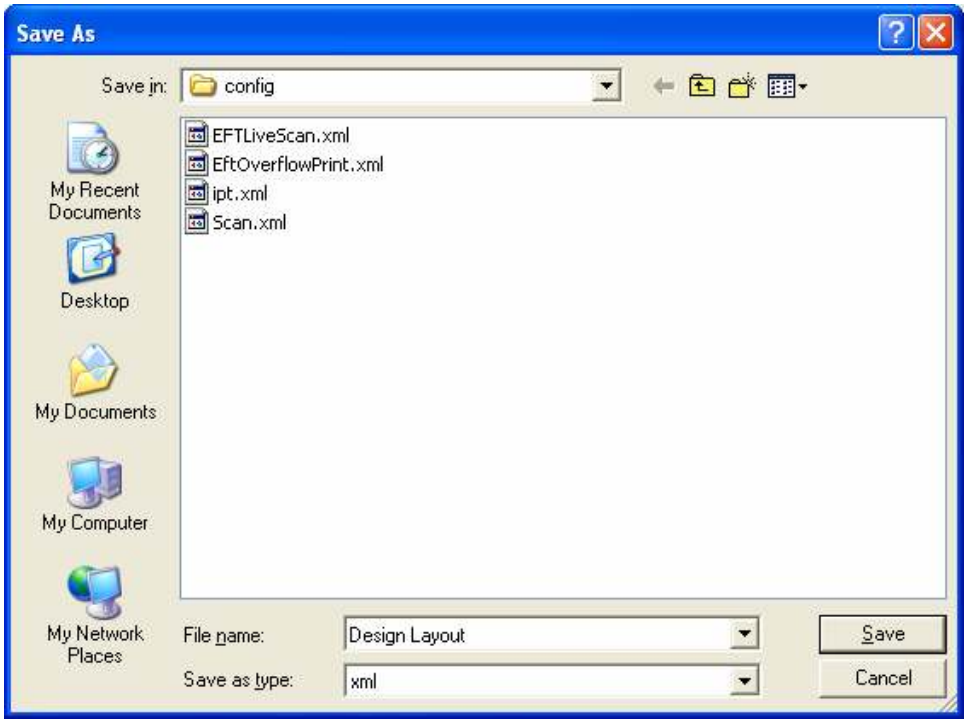

Figure 2.15 Save to disk prompt.

2.3.3 Editing an existing layout

If the user wishes to change the screen layout at any point then the user clicks in the open menu from the Menu bar. Then the **XML Generator** prompts the user to select the XML file that has the settings of the previous created layout and click **Open.**

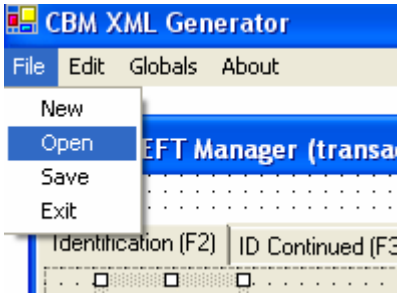

Figure 2.16 Open Menu item.

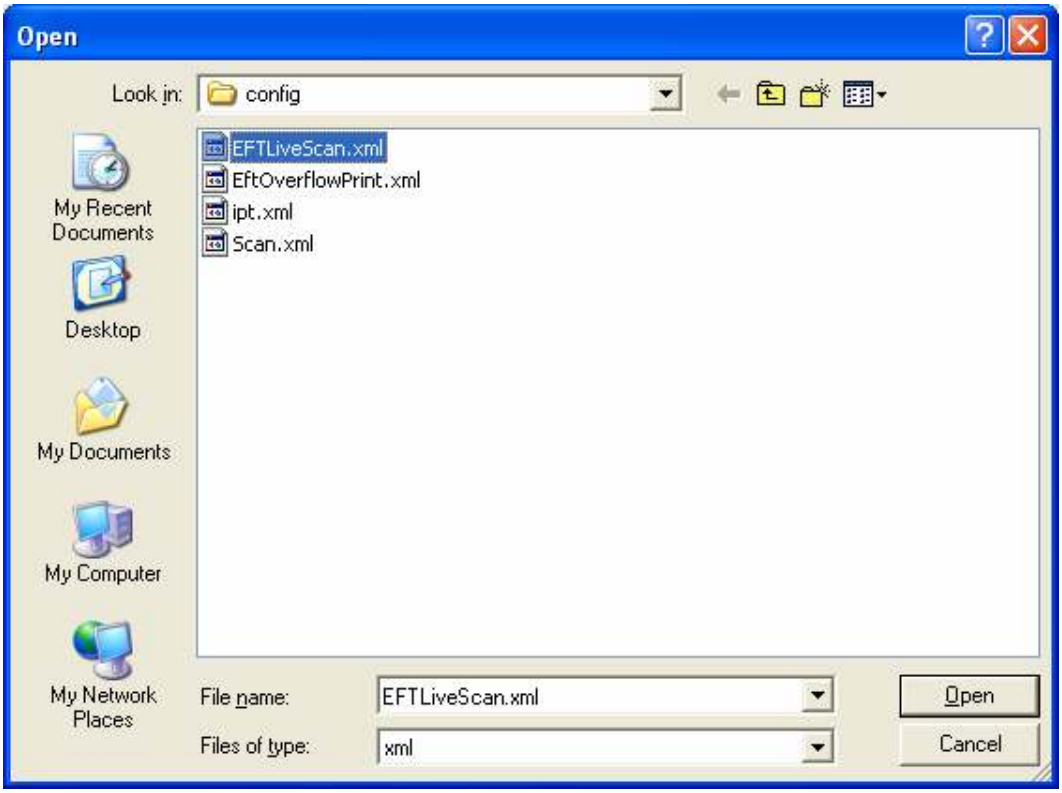

Figure 2.17 Open from disk prompt.

Once the layout is selected, the application creates the layout from the XML and displays it as shown in Figure 2.18. The control properties can then be modified or added easily.

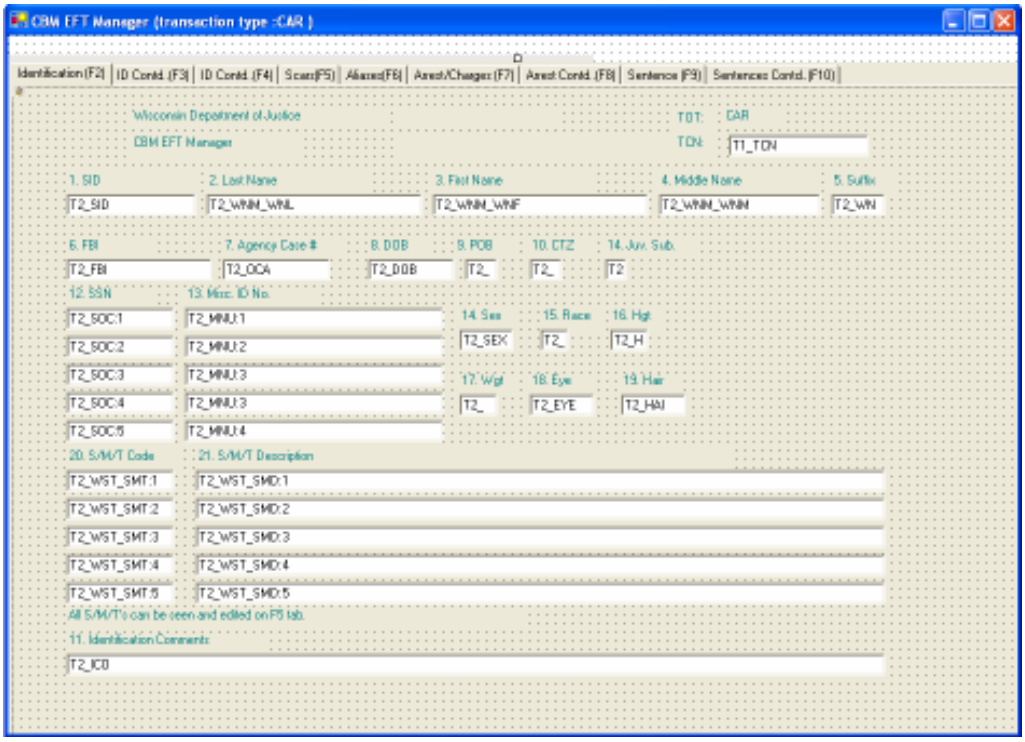

Figure 2.18 Screen designed from XML file.

As shown in Figure 2.19, if a user wishes to delete a control then the control should be first selected and then the **Delete** button should be clicked from the Edit menu. Incase a control was accidentally deleted then the **XML Generator** application can repeat the previous step for the user by clicking on the **Undo** button from the Edit menu as shown in Figure 2.20. Similarly as in Figure 2.21, the layout can be completely cleared by selecting the **Clear** option. Finally the application can be exited by clicking on the **Exit** option as shown in Figure 2.22.

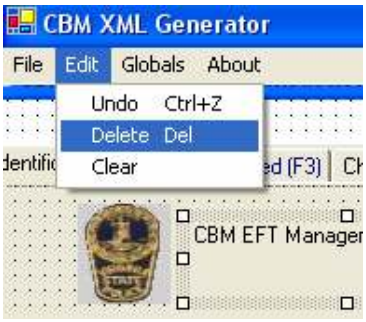

Figure 2.19 Delete Menu item.

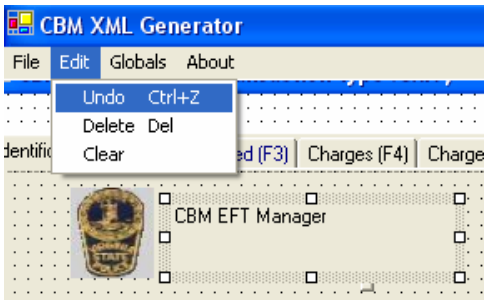

Figure 2.20 Undo Menu item.

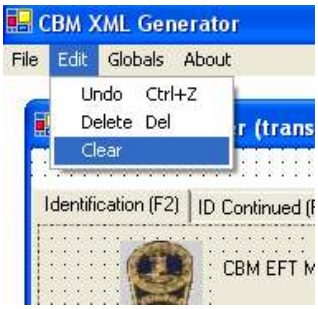

Figure 2.21 Clear Menu item.

Figure 2.22 Exit Menu item.

3. SYSTEM DESIGN

3.1 Systems Used In the Project

Microsoft SQL Server 2000 Evaluation Edition and Developer Edition were used as the back end database for this application. The application was developed using Visual Studio.NET 2003 and .NET framework 1.1. VB.NET programming language was used to design and code all the functionality for the application. The development environment was set up on a Windows XP professional operating system. Table 3.1 shows the requirements of the system that was used for developing this application. [Microsoft 2004]. Microsoft SQL Server 2000 Desktop Engine (MSDE 2000) was also installed on the development environment.

3.2 Systems Needed for Normal Use of the Application

This application can be successfully deployed on any system that has a Microsoft SQL Server 2000 and .NET framework 1.1 Redistributable Package Installed. The system requirements for running this application are shown in Table 3.1 [Microsoft 2004]. Since the application was entirely developed using latest technology like .NET, the application can reliably run in the latest version of the operating systems such as Windows XP or 2000.

Table 3.1 Systems Needed for Normal Use and developing of the Application

Minimum Requirements	
Processor	Intel Pentium or compatible 166-megahertz (MHz) or higher processor
Operating System	Windows XP Professional or Windows XP Home Edition
Memory	Home Edition: 64 megabytes (MB) of RAM; 128 MB recommended
Hard Disk	Enterprise, Standard, Evaluation, Developer, and Personal Editions require: 95–270 MB of available hard disk space for the server; 250 MB for a typical installation. The application requires: 660 MB of hard disk space, 190 MB additional hard disk space required for installation (850 MB total)
Drive	CD-ROM drive
Display	VGA or higher-resolution monitor
Software	.NET framework 1.1 redistributable, SQL Server 2000 Evaluation Edition and Developer Edition

3.3 Database Design

The database was designed in SQL Server 2000. SQL Server 2000 has role memberships for login and user security accounts, which is responsible for permissions / authentication procedures in order to allow various tasks both for server and for the database. Some roles are fixed, i.e. specified by SQL Server. When the user connects to

the SQL Server database through the VB.NET application, they must identify themselves through the security accounts. The application then performs a look-up at the SQL server for user rights. There is a table called *user settings* that has access details and another table called data fields table that has the actual data.

3.3.1 User Settings

The fields that are used in the user settings table are *user, accessrights, login, and password.* U*ser* is the description of the user in the domain and *accessrights* is given two options: read and write. If the *accessrights* is set to read, then the user can only view the record and not make changes to the file. If the *accessrights* is set to write, then the user can have full control of the file. When the user opens the file, all the controls are set to read only. The user is required to enter the login and password assigned to him by the administrator in order to edit or change the data in the fields. The *login* and the *password* fields have the access information of a user.

3.4 Data Flow Diagram

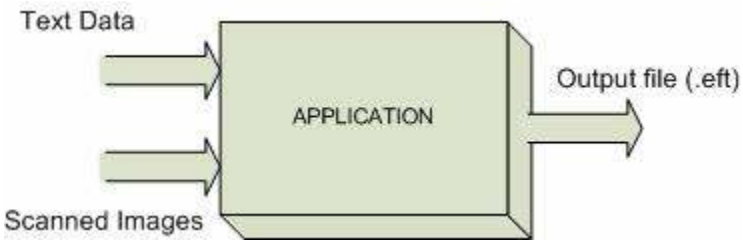

Figure 3.1 Block Diagram of the Application.

As shown in Figure 3.1, the application captures all the entries that are keyed into the data entry fields. The images needed for a particular transaction are also captured. When the user decides to save the transaction, first the fields are validated, and then scanned images and the fields are saved in the database. The file is saved in a format called electronic format (.EFT).

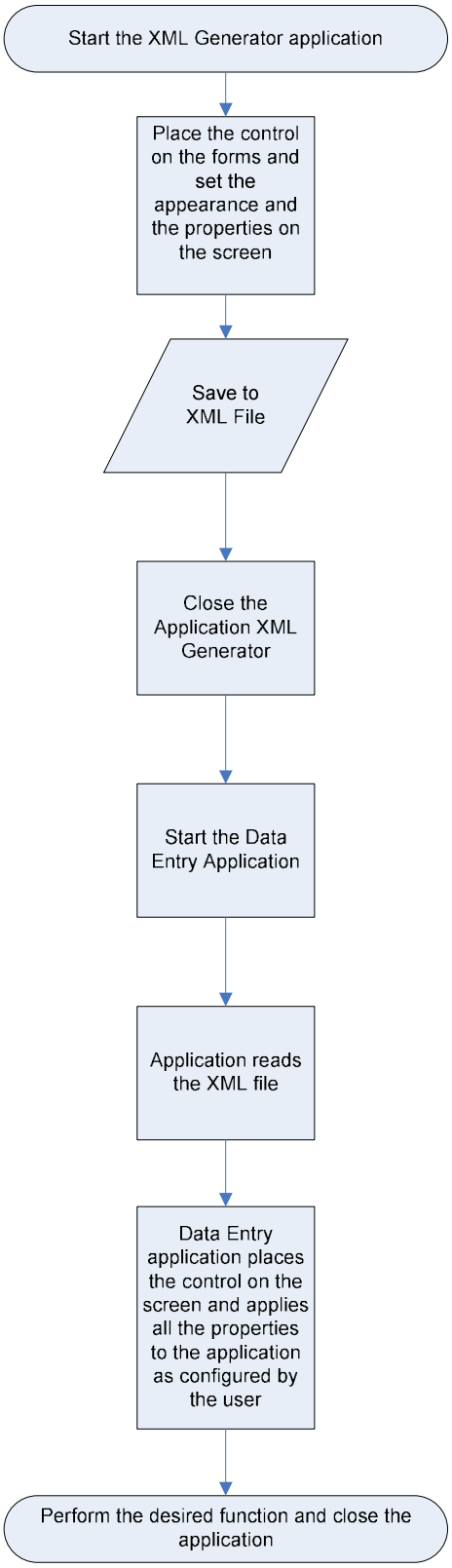

Figure 3.2 Flow Chart of the Application Process.

As shown in Figure 3.2, the **XML Generator** application is started by running the executable file. Then the user can place the controls on the form by dragging and dropping them from the tool bar. The height, color and position of these controls on the screen can be easily adjusted. The user can also set the properties of the field on the form. Then the user can save the fields on the screen and the functionality that he/she selected by saving it to an XML file. The settings can then be saved and closed.

When the executable is run again, the application reads the XML file saved by the user and automatically place the controls and add all the functionality selected by the user. The user can then have access to the fully functional system. If at a later time the user wishes to modify a control on the screen, the application screen can be modified and saved back to the XML file.

3.5 File Formats

XML Generator application saves the all the properties set on the user interface screen in an XML (Extensible Mark up Language) format. Since XML is a simple, very flexible text format, it is playing an increasingly important role in the exchange of a wide variety of data [W3C 2004]. Also the XML files can be easily viewed using tools like Internet Explorer and Notepad. Further the XML file can be edited using Windows Notepad which comes integrated with Windows Operating systems. This ensures that the application can be used and moved across all windows operating systems with ease.

4. EVALUATION AND RESULTS

The system is dynamic and configurable, so it was very important to test all the features of the application to see if it performed the desired functions reliably and effectively. Since this system was aimed to be used by various organizations, it was very important that the information is saved and stored reliably. The major effort was delegated to test the usability of the system, security, and stress testing. At each step the system's program was verified for its actual purpose and final results.

4.1 Usability Testing

Unless the users of the system feel comfortable with the system, the project would never be a success. Much effort was spent on observing the ease of use for users with varying knowledge or education. Most of the data-entry personnel today have very little formal education. It was highly important that the program was easy to grasp by users of varying capabilities and levels of education and experience. Feedback and suggestions were taken regarding the interface design and usability of the system. Their feedbacks were rated on a scale from Excellent to bad. The module were later deployed in a test environment that had users of different level of education and experience for at least one month to get a healthy sample space of feedback that was the true representative of the application's user-friendliness. The feedback and suggestions were then used to make changes to the functionality and look of the application until the users were fully comfortable with it.

A comprehensive training and demonstration of the application was given to all the users testing the application. Each user was given an opportunity to get familiar with the system during the training session. They were then given a common task of building a

data entry screen from a sketched layout using the **XML Generator** application and a detailed user guidance manual was provided for their guidance. At the first stage, the applications were deployed on fourteen user's workstation and were given ten days to thoroughly test the system and give their feedback. Table 3.2 illustrates some of the key comments obtained from the users.

Table 3.2 User Feedback during Initial Interface and Usability Testing

User #	Background of the Users	User Rating	Key Comments
1	Undergraduate	Good	Accessibility option was not provided.
2	High School	Excellent	Version information was not provided
3	Masters Degree	Good	The Application should remember its screen position the last time when it was closed and open at the same place.
4	High School Drop out	Good	The control should be drag and drop like controls.
5	Associate Degree	Excellent	Frequently used functionality like align, color, bolding text should be provided in the Menu bar
6	High School	Excellent	The navigation in the application should be made easy to use only with the Keyboard.
7	High School	Excellent	Hot Keys should be provided for key functions like New, Edit and Save.
8	Associate Degree	Good	The window should have scroll bar if the application is not resizable beyond a point.
9	High School	Good	Tabs should switch faster
10	High School	Good	The application should warn for error before saving and not while changing tabs.
11	High School Drop Out	Excellent	Hot Keys or Keyboard shortcuts should be provided
12	Associate Degree	Excellent	Should have help functionality with the application
13	Undergraduate	Excellent	Tool bar tip should provide details of the control.
14	Masters Degree	Good	Panels should be provided to group similar controls in the application

After the initial feedback was provided by the users, the requested enhancements were added to the application. The users were then given training on the new features that had been added to the system and the enhanced application was deployed again for the

users to test for another month. Since the users had already gone through a thorough testing the first time, the enhancements added to the system from their initial feedback enabled the application to fit most of their expectation the second time. The Usability testing proved to be a success as it got a very good feedback from people of all educational backgrounds considered for the test.

4.2 Performance Testing

The timing of performance testing depends on the model or methodologies being used [Henninger 1998]. Once the system was tested without fault with a single user, a load of concurrent users were simulated to test key issues of whether or not the application crashes due to continuous data locks, etc.

Microsoft Application Center Test (MACT) was used for testing the performance of the **XML Generator** Application since the tool comes integrated with Microsoft Visual Studio.NET 2003. MACT enables you to gather performance information and makes capacity decision on the .NET applications. The tests can be created to simulate several users simultaneously requesting pages for an application. These simulations help determine the speed, stability and responsiveness of the application.

The most important aspect that was considered for testing was the application's interaction with Database. The user interacts with the database only for accessing the user name and password, and other configuration information for accessing the application. So the test cases were written to test connection between the test (the client) and the SQL Server.

Table 3.3 illustrates the results of the performance testing done on the application against the SQL Server. Some of the parameters displayed in the table 3.3 are explained below:

- Test Name- Name assigned to the specific test.
- Test Started- Specifies the time at which the test was started.
- Simultaneous Browser connections- The number of connection made by the application at one time.
- Warm up time (secs): Time taken for the initial operations by the XML generator before performing the required task. This value could be deducted to calculate the actual duration of the test.
- Test Duration – Denotes the time period taken by MACT for completing the requested task.
- Test Iteration- Specifies the number of database connection made in the test.
- Test Type- The value of Dynamic shows that the operation required no manual intervention and the test was conducted automatically.
- Total number of Requests- Actual count of the number of database requests triggered by MACT.
- Total number of connections: Number of connection successfully made by the application. The value should be close to the total number of requests made.
- Average requests per second: It is the average of the time taken by the application to make each connection, complete the operation and disconnect.
- Number of unique requests made in test - This denotes number of unique connection by the SQL server when it was called by many applications.

- Errors Counts- Displays the errors that were generated by various protocols like HTP, DNS and sockets while running the test.
- Average Bandwidth- Specifies the average load applied on the Network during the test.
- Number of timeout errors: Time out error occurs if the SQL server takes a long time to complete the operation. This parameter displays the timeout errors that resulted from the database test.

Table 3.3 Results of Performance Test using Microsoft Application Center Test

Test Name:	XML Generator Test
Test Run Name:	Report- XML Generator Test –June 20, 2005 15-20-34
Test Started:	6/20/2005 3:15:32 PM
Test Duration:	00:00:05:00
Test Iterations:	34,028
Test Notes:	-
Properties	
Test type:	Dynamic
Simultaneous browser connections:	1
Warm up time (secs):	0
Test duration:	00:00:05:00
Test iterations:	34,028
Detailed test results generated:	Yes
Summary	
Total number of requests:	34,029
Total number of connections:	34,029
Average requests per second:	113.43
Average time to first byte (msecs):	5.74
Average time to last byte (msecs):	5.94
Average time to last byte per iteration (msecs):	5.94
Number of unique requests made in test:	1
Number of unique response codes:	1
Errors Counts	
HTTP:	0
DNS:	0
Socket:	0
Additional Network Statistics	

Average bandwidth (bytes/sec):	619,894.95
Number of bytes sent (bytes):	11,842,092
Number of bytes received (bytes):	174,126,393
Average rate of sent bytes (bytes/sec):	39,473.64
Average rate of received bytes (bytes/sec):	580,421.31
Number of connection errors:	0
Number of send errors:	0
Number of receive errors:	0
Number of timeout errors:	0
Response Codes	
Count:	34,029
Percent (%):	100.00

4.3 Discussions

When the application was demonstrated to Computer Science experts from Texas A&M University-Corpus Christi, the following question was raised during the discussion:

- If there are multiple users modifying or building a user interface at the same time using the XML Generator application and they add identical field to the design surface, Will the application accept duplicate entries when they try to save?

  Answer: When one of the users completes the user interface and saves the screen layout, a database column is created for every field that is added to the application. Now when the other users try to create and save a field that is already mapped to a column in the database, the **XML Generator** checks to see if the field has already been created and does not allow the user to save the duplicate field. Knowing that the user is not aware of the fields created by another user at the same time, the user is notified of the update and the screen is refreshed and updated with the fields created by the other user.

- If an application is developed using other Tools, can XML generator change its interface? Or do you have to start it from scratch?

  Answer: **XML Generator** will only be able to modify user interface designed by it. Any user interface designed using other tools will not be recognized by the application. To achieve that, the user will have to design the screen from scratch using the XML Generator application.

5. FUTURE WORK

Currently the **XML Generator** provides most of the components needed for helping the user design a fully functional data entry application. Some of the discussions with the experts revealed that a search feature would be a valuable addition to the XML Generator application. Hence a Graphical user interface will be built in to the application to allow the users to easily query and search from the saved records in the database. In future more components such as combo box, date time picker, Group Box and Panel will be made available for the user. Also the current .NET framework is available for C#.NET in LINUX. In near future it can be expected that LINUX will support .NET framework for VB.NET programming language too. At that point the application can be deployed across LINUX operating systems.

6. BIBLIOGRAPHY AND REFERENCES

[Althoff 1998] Althoff, K.D., Bomarius, F. and Tautz, C. Using Case-Based Reasoning Technology to Build Learning Software Organization. *Interdisciplinary Workshop on Building, Maintaining, and Using Organizational Memories*, ECAI-98.

[Ambler 1995] Ambler, S. *Use-Case Scenario Testing. Software Development*, vol. 3, no. 6, (July. 1995), 53-61.

[Balena 2002] Balena, F. *Programming Microsoft Visual Basic.NET,* 2002.

[Ellmer 1996] Ellmer, E., Merkl, D., Quirchmayr, G. and Tjoa, A.M. Process Model Reuse to promote organizational learning in software development. *Proceedings of the 20th International Computer Software and Applications Conferen*ce *COMPSAC96*, (Aug. 1996), 21-26.

[Francis 2002] Francis, N. *Professional XML for .NET Developers*, June 2002

[Henninger 1998] Henninger, S. An environment for reusing software processes. *Proceedings of the 5th International Conference on Software Reuse*, (Aug. 1998), 103-112.

[Little 1998] Little, M. Building Configurable Applications. *Proceedings of the 4th IEEE International Conference on Configurable Distributed Systems*, (May 1998), 6-10.

[Microsoft 2004] Microsoft .NET Framework Development Center. Available from http://www.microsoft.com/sql/evaluation/sysreqs/2000/default.asp (visited Nov.14, 2004).

[Myers 2003] Myers, B. Creating complete user interfaces by demonstration. *Proceedings of the SIGCHI conference on Human factors in computing systems*, Amsterdam, The Netherlands, (April 2003), 293-300.

[W3C 2004] World Wide Web Consortium. Available from http://www.w3.org/XML/ (Visited May 2005).

7. ACKNOWLEDGMENTS

I would like to thank Dr. Long-zhuang Li (Project Chair), Dr.Ajay Katangur (Project Committee Member) and Dr. Scott King (Project Committee Member) for accepting the proposal to implement **XML Generator** project. I would also like to thank Mr. Michael Lord from CBM Archives for being cooperative in providing all information required for this project.