Summer 2006

Using an Object-Oriented Approach to Develop a Software Application

Paul Duvall
Regis University

Regis University
School for Professional Studies Graduate Programs
Final Project/Thesis

Disclaimer

Abstract

This paper describes a software development project completed using an object-oriented approach. Because Visual Basic .NET was used to build the Windows-based application, application design patterns were identified that would be beneficial in this development environment. A discussion of which design patterns were selected and how they were implemented using Visual Basic .NET is included, along with descriptions of object-oriented design documents and their role in the project. The paper also provides details about how the project has and will be used to provide teaching examples in software development courses.

Table of Contents

List of Tables

List of Figures

Chapter 1

1.1 Introduction

Two questions were instrumental in the selection of this project. First, what object-oriented design patterns are practical to use in a Visual Basic .NET application? Second, what relevant teaching examples can be derived from an object-oriented project? To help answer these questions, this paper will review how a software development project was created using an object-oriented methodology. It includes a description of the different project phases and their implementation in the project's lifecycle. A review of the project and potential areas for improvement are also discussed.

1.2 Project Background

The author helped to develop an assessment to determine the computer aptitude of prospective students. One of the deliverables of this project was a software application that tests proficiency with common computer related tasks including keying, working with e-mail, opening applications, and using the Internet. This technology skills assessment helps place students in the appropriate Information Technology courses.

One challenge in selecting a professional project was to identify a project that fulfilled the course requirements and also served a useful role beyond the class. The project

selected also needed to be one that could be used by the author in teaching object-oriented design. This led to the creation of a project for the development of an administrative add-on for the skills assessment. The add-on would provide options to update the test's content and cutoff scores, while the project documents would be used as examples to supplement textbook materials. Example project documents include use cases, diagrams, and source code

At the completion of the project, the administrative add-on was tested to ensure that it accurately updated the assessment's content. It is currently being evaluated for implementation. While the add-on is not currently available to the test administrator, project documents have been reviewed and discussed in software development classes the author facilitates.

1.3 Project Relevance

One goal of the project was to provide a convenient way for the test's administrator to update the content and cutoff scores for different sections of the assessment. The add-on uses a graphical user interface that provides access to the content and scores related to the different test sections. This enables the administrator to review and update all content areas of the skills assessment, easing the process of implementing changes to the assessment.

Another goal for the project was to create an example of how to apply object-oriented design patterns in a Visual

Basic .NET project. Design templates developed and tested in this project will provide a framework for future projects. Using design patterns in a .NET application will make it easier to identify the benefits they offer the .NET developer.

The design process will also supply several illustrations for software development courses. For example, students will be able to review the design requirements presented by the use cases and class diagrams. The source code from the project will show how these requirements were realized. Reviewing the discussion of project's implementation and the lessons learned from it will offer another learning opportunity for the students. A final example is showing the students how the project's data requirements translated into the physical database design.

1.4 Project Issues

Two critical issues had to be addressed in the development of the project. These were determining what design templates to use and identifying how to modify the skills assessment to use the content and scores maintained by the administrative add-on. The following paragraphs describe how these issues were addressed.

The design templates that would be used in the project had to be specified before the coding began. To determine which ones to include, Martin Fowler's book *Patterns of Enterprise Application Architecture* was used. A description

of the patterns selected is included in Chapter 2, with a discussion of the implementation details included in Chapter 4.

Identifying the changes required to allow the skills assessment to work with the add-on was an important aspect of the project. The technical feasibility of the project depended on the complexity of this integration. If significant modifications to the skills assessment application were required, then it would be better to rewrite it instead of building the add-on. After reviewing the skills assessment application, it was determined that the integration between it and the add-on could be achieved through sharing the add-on's data source and retrieving the data elements it maintained into the appropriate test content sections.

1.5 Project Discussion

The project required research into which design patterns were relevant to a Visual Basic .NET application. In the case of the domain layer, there were multiple patterns that had to be considered. These options were the Transaction Script, Table Module, and Domain Model (Fowler, 2003, p. 96 – 97). The next paragraphs give an overview of these choices, and the pattern selected is identified in chapter 2.

The Transaction Script model is the simplest one to implement. It separates each request for data into a

transaction. This works well with a relational database, but can cause code to be repeated when similar access is required by different transactions.

Another option was the Table Module. This model is not as simple to implement as the Transactions Script, but is not as difficult as the Domain Model. It works well in a development environment that provides additional objects for manipulating data.

The final option was the Domain Model. A strength of this model is its ability to handle difficult business logic. However, it is the most challenging to implement.

1.6 Project Scope

The project's focus was to create the add-on using object-oriented development methods. Only the modifications that were required to allow the skills assessment application to access the test content maintained by the add-on were considered. The changes to the skills assessment allowed it to retrieve content settings maintained by the add-on and display them without user intervention. This was accomplished by giving the assessment application access to the database used by the add-on to store the test content updates. The project used Visual Basic .NET to develop the add-on, and took an object-oriented approach to designing the add-on. Testing activities, including code review and unit testing, were used throughout the development process to verify that the requirements identified by the use case

scenarios were fulfilled. At the project's conclusion, the initial version of the administrative utility was available for implementation. A deployment project built in Visual Basic .NET was created to provide an automated installation of the application.

1.7 Project Terms

There are several software development and project specific terms used throughout the project paper. Table 1 includes a list of these terms with their definitions.

Table 1 – Project Terms

Project Term	Definition
Class diagram	A visual presentation of the attributes, methods, and relationships among the classes used in the project.
Data dictionary	A database design tool used to identify the attributes of the columns used in the database tables.
Entity relationship diagram	A visual representation of the relationships between the database tables.
Integrated development environment	The Microsoft Visual Studio development environment used to create the administrative tool.
Microsoft Project	A Microsoft software application used to manage the project by tracking the tasks needed to complete the project.
Use case	A modeling tool that describes the interaction between an actor and the application being designed.
Visio	A Microsoft software application used to create the various diagrams during the application's design.
Visual Basic .NET	A Microsoft software development tool used to build the administrative add-on.

Chapter 2

2.1 Research Overview

Several resources were used to assist in the development of the project. Areas of the project that required research included application design, specific code techniques, and database design. A discussion of the resources used that addressed these areas is included in the next three sections.

2.1.1 Application Design Research

Martin Fowler's book *Patterns of Enterprise Application Architecture* provides several examples of design patterns that can be used in the development of an application. *Object-Oriented & Classical Software Engineering, Sixth Edition* by Stephen Schach discusses the stages of developing an application from design to implementation. Because it includes a discussion of both classical and object-oriented techniques, it makes it easy to identify the differences between these two methodologies. It also includes recommendations for testing and source code standards.

2.1.2 .NET Code Techniques Research

When developing the source code for the project, it was important to understand how using certain objects and code techniques will influence performance. The *MCAD/MCSD Developing and Implementing Windows-based Applications with Visual Basic .NET and Visual Studio .NET* book by Mike

Gunderloy includes several examples of best practices to increase the performance of a Visual Basic .NET application. It also contains systematic descriptions of how to implement different types of Visual Basic .NET projects.

2.1.3 Database Design Research

Storing and retrieving data was an important aspect of this project. *Database Systems: Design, Implementation, & Management, Sixth Edition* by Peter Rob and Carlos Coronel contains information on how to design and implement a database. A strong point of this resource is that it provides examples and descriptions of different database design techniques.

2.2 Patterns

An area of the project that needed to be addressed was determining which patters should be used in a Visual Basic .NET windows application. In his book *Patterns of Enterprise Application Architecture*, Martin Fowler discussed several patterns that could be used in software development. The book includes both a description of the pattern and code examples of how it would be implemented in a project. One difficulty with this resource was that the majority of the code examples were written in Java, and the project was created using Visual Basic .NET. However, Martin did include some discussions of how particular patterns would be useful in .NET applications. Using this information, along with a

basic understanding of Java, helped in identifying how the patterns could be included in the project.

There are three principal layers used in developing an application (Fowler, 2003, p. 19 – 20). These include the presentation, domain, and data source layers. Characteristics of these layers are discussed in the next three sections.

2.2.1 Presentation Layer

The presentation layer provides a way for the user to interact with the application (Fowler, 2003, p. 19 – 21). For example, the Windows forms in this project include buttons and menu items that the user can select by using the keyboard or mouse. By clicking the Change Password menu option, the user generates an event procedure that causes the application to display the Change Password Windows form. This form allows the user to select a new password.

The project's software developer used the Model View Controller (MVC) pattern to implement the presentation layer. The model represents the domain logic of the application, the view focuses on displaying the information requested by the user, and the controller is used to handle the user's interaction with the interface (Fowler, 2003, p. 330 – 332). It is important to separate the user interface components (view and controller) from the model. A discussion of how this was accomplished is included in chapter 4.

2.2.2 Domain Layer

To function correctly, a software application has to implement the use case requirements and ensure that exceptions are handled correctly. The business logic is implemented by the domain layer (Fowler, 2003, p. 20). Because business logic describes how the application should handle different data elements, there is a strong connection between this layer and the data source. However, it is critical to incorporate a mechanism that effectively separates these layers to make the application easier to maintain.

For this project, the domain layer was implemented through the Table Module pattern (Fowler, 2003, p. 125 – 128). This pattern was selected because it works well with ActiveX Data Objects (ADO). ADO is used in Visual Basic .NET applications to provide access to different data sources. To utilize the pattern, the developer created a series of classes that correspond to each database table required by the application. Each class included the methods required to process the data stored in the associated database table.

2.2.3 Data Source Layer

The data source layer is responsible for communicating with the database (Fowler, 2003, p. 20). This layer is used to retrieve data requested by the application. It can also track changes made to the data by the application and update the database with the modifications.

Because ADO is the data access technology used in .NET development, it was used to incorporate the Record Set pattern into the application (Fowler, 2003, p. 508 – 510). This pattern takes advantage of ADO by using Dataset objects. Datasets can include multiple rows of data from one or more tables in the database.

2.3 Object-Oriented Design

The object-oriented methodology makes use of several different models to communicate the design of the system. Because they define the key components and requirements of the system, the developer uses these models to produce the software product. Important elements of the use case, class diagram, and sequence diagram will be reviewed in the following three sections.

2.3.1 Use Case

To create the software product, the developer needs to know how the user will interact with the application, what the outcomes of the typical interaction will be, and what error conditions might be encountered during the process. The use case describes what actions the user will take and how the application will respond (Schach, 2004, p. 349 - 350). To begin developing a list of use cases, the different tasks required by the application were defined. These unique tasks became the names for the use cases. The external entities, referred to as actors, that interact with the application were also identified.

The steps required to complete each use case successfully provided the requirements for the software design. Alternative paths that the user could take and the application's response were included to define the error handling routines that needed to be developed.

2.3.2 Class Diagram

Identifying the classes needed to meet the requirements of the use cases and the relationships between them is another critical element of object-oriented design (Schach, 2004, p. 365 – 370). The class diagram lists the attributes and methods for each class needed by the application. Attributes are data items that define characteristics of the class. Methods are actions provided by the class. The developer sets and returns the attribute values, and calls the methods to perform actions on the class.

2.3.3 Sequence Diagram

The sequence diagram illustrates how the use case will be implemented in the code. They do this by showing the order that methods are executed on the various classes involved in the application. Lifelines show when a class is instantiated and destroyed. Returned messages and objects are also shown in the diagrams. The sequence diagrams described by Stephen Schach (2004, p. 377 – 381) provide a good illustration of the elements included in this design model.

2.4 Application Development Practices

In developing an application, it is important to establish coding standards, develop guidelines to improve the application's performance, and define a testing methodology for the application developers. Establishing and following code standards throughout a project will make it easier to maintain the application. Paying attention to performance at the start of development will help prevent costly rewrites to correct these issues prior to implementation. Defining a testing methodology for the software development team reinforces the concept that testing is a continuous process, and not a task only completed at the end of the project's life cycle. Research used to develop these practices will be described in the following sections.

2.4.1 Coding Standards

Implementing coding standards provides a framework for the coding process. Developers frequently have to review and modify code written by another team member. Having standards in place helps decrease the amount of time needed to understand what a particular section of code is doing. One coding standard implemented in this project was including a comment at the top of each developer-defined procedure or function that identifies its purpose, the input parameters, and the data it returns.

Using meaningful names and laying out the code to make it easier to follow were other coding guidelines used during the application's development (Schach, 2004, p. 434 – 437). One way meaningful names were used was that the variable name was prefixed with three characters that help identify its data type. For example, str was used at the beginning of a String variable, and int was used to identify an Integer variable. This naming convention helps a developer understand what values a variable can store without having to spend time searching for the variables declaration statement. A code example from the project that illustrates the implementation of these standards is included in chapter 4.

2.4.2 Application Performance

Application performance is a critical issue for a software product. It seems that users will always notice and report when an application is performing slowly. Depending on the type of application being developed with Visual Basic .NET, there are specific techniques that the developer can use to improve the application's response. Two techniques used in this project are determining when to throw exceptions and using the DataReader when an updateable dataset is not required (Gunderloy, 2003, p. 932 – 936).

Throwing exceptions is a costly operation in Visual Basic .NET. There are times when an exception needs generated to pass on error details to the developer.

However, there are situations where error information could be conveyed to the developer through the return value of function.

There are also times where the application does not require an updateable dataset. For those situations, the developer should use a DataReader to return data from the database. The DataReader is a read-only, forward-only collection of rows from the table, making it a fast way to read data.

2.4.3 Testing

Several techniques are available for testing an application. These include black-box testing, glass-box testing, and code walkthroughs (Schach, 2004, p. 452 – 458). Black-box testing uses the requirements to design the tests for the product. Glass-box testing looks at the code and attempts to test the individual statements to make sure everything is functioning correctly. During code walkthroughs, developers review the source code to identify logic errors.

2.5 Database Design

Data was a critical element in this project. The ability to retrieve and update screen text elements without modifying the online technology skills assessment application was a primary goal of the project. A major task in developing the logical database was normalizing the design to ensure it would be effective.

2.5.1 Normalization

Normalization is an important part of database design because its implementation eliminates data redundancy (Rob & Coronel, 2004, p. 186 – 188). Data redundancy refers to multiple tables in the database storing the same information. This presents a problem when data is inserted, updated, or deleted because the developer has to make sure each of the tables containing the data is updated successfully.

Although there are five normalization forms, only the first three were used in this project. The reason for this is that most business applications only require the database to be in third normal form (Rob & Coronel, 2004, p. 184). A general discussion of the three normal forms is included in the next paragraphs.

The first normal form involves defining the key attributes for the table, ensuring that the non-key columns depend on the primary key, and eliminating repeating columns from the table (Rob & Coronel, 2004, p. 188 – 191). At this point, the primary key for the table is set. The primary key uniquely identifies each row in the table. Even though the non-key elements display a dependency on the primary key, some may depend on only part of the key or may also depend on other columns in the table. These dependency issues will be resolved as the table moves to the second and third normal form.

Removing the partial dependencies from the table is accomplished through the second normal form (Rob & Coronel, 2004, p. 191 – 192). Columns that exhibit partial dependencies only rely on part of the primary key. Because of this, partial dependencies can only occur in tables where the primary key is composed of multiple columns. If a table does not have a composite key and is in first normal form, it is also in second normal form.

Third normal form addresses the concept of transitive dependencies. This type of dependency is present when columns depend on other non-key columns in the table (Rob & Coronel, 2004, p. 192 – 194). To correct this issue, these columns and the non-key column they depend on are removed to become a new table.

2.6 Project's Contribution

One way this project will continue to contribute beyond its current implementation is by providing templates for future development projects. There are numerous development examples throughout the project. Reviewing how the application design patterns were implemented will make it easier to include them in future projects. The format used to produce the use cases with their success and alternative scenario descriptions will be followed in future development activities. Another template is the data requirements document. Included in the data requirements are the columns that need to be defined, a table definition that includes

its name and columns, and an entity relationship diagram to show the relationships between the tables.

A second way this project will be beneficial is as a teaching tool. The author currently instructs software development courses. An effective method of teaching software design is to provide relevant examples to the students. This enables to students to visualize the process being discussed. The project and its artifacts will provide teaching illustrations that will be useful in several classroom discussions. Elements of this project will be used to illustrate software development practices, database design techniques, and general project management methods.

Chapter 3

3.1 Research Overview

The project needed to identify object-oriented design patterns that could be applied to a Visual Basic .NET project, and provide design and code examples that would be useful in teaching software development concepts. To help address these issues, the project applied concepts that were taught in software design courses at Regis University. An overview of the methods used in the project is included in the following paragraphs.

A project life cycle that included analysis, design, testing, and implementation phases was developed to guide the project. Each phase had deliverables, like design documents or a version of the software, that were produced during that project stage. There were milestones for each phase that signaled the completion of the phase's tasks.

Several software products were used during the project. Microsoft Project was used to identify the project's tasks and track their progress. Microsoft Visio provided the drawing tools needed to create the design documents, and Microsoft Word combined the use cases and drawings into a design document.

3.2 Life Cycle

The project's life cycle is similar to the Microsoft Solutions Framework model (*Microsoft IT Solutions: MSF*

Process Model v. 3.1, 2002). This model includes the envisioning, planning, developing, stabilizing, and deploying phases. For this project, the life cycle included the analysis, design, testing, and implementation phases. The primary difference between the models is that the project's model combined the tasks from envisioning and planning into the analysis phase.

Several tasks needed to be completed during each of these phases, and there were target completion dates for each phase. However, tasks in different phases could be worked on simultaneously. For example, an initial class diagram was developed during the analysis phase, but changes to the diagram were implemented during the design phase. The phases provided a guideline for completing the tasks required by the project. Each phase is discussed in the following four sections.

3.2.1 Analysis Phase

The focus of the analysis phase was to build the blueprint for the remainder of the project. To accomplish this goal, several artifacts were developed. Three key artifacts generated from this phase were the use cases, class diagrams, and sequence diagrams.

Use cases were developed to ensure an understanding of the application's requirements. The first step was to identify the key tasks that the application needed to accomplish. Once these were identified, a detailed list of

actions taken by the actor and application that would successfully complete the use case was developed. In addition to the success scenario, the use cases included a description of alternate interactions between the actor and application. The detailed use case descriptions provided the requirements to complete each task needed by the application.

After the use cases were built, class diagrams were developed. Methods and properties assigned to the classes matched the requirements of the use case. For example, the User class included properties to store the data elements defined by the use cases that required the User class. Methods supplied by the User class fulfilled the functionality needed by these use cases.

Sequence diagrams combined the details of the classes and use cases to show the communications required between classes. To build the sequence diagrams, the classes needed by the use case were identified. Next, the order that functions were called in each of the classes was specified by adding them to the diagram. Communication arrows with message descriptions showed the interaction between the classes.

3.2.2 Design Phase

Once the analysis artifacts were reviewed for completeness, the coding of the application began. Key objectives for this project phase were to verify technical

feasibility and use the artifacts identified during analysis to develop the application. The remainder of this section will discuss how these tasks were implemented in the project.

The key to the technical feasibility of the project was to determine if the skills assessment application would be able to use the content and scores provided by the add-on tool without completely rewriting the application. After determining how the content and score data would be stored, the original application was reviewed. After the review process, it was determined that with a few simple modifications, it could use the output from the administrative tool.

The administrative add-on was built by using a combination of the use case descriptions, class diagrams, and sequence diagrams. Since the steps to fulfill the requirements of the application were included in these artifacts, developing the software using these items helped ensure that the requirements were fulfilled. The data tables for the application were developed and added to the database.

3.2.3 Testing Phase

Testing was a critical component of the project. Several different testing methods were required during its lifetime. Document review was used during the analysis to determine if steps were missing from the use cases. During

coding, sections of the code were stepped through to ensure completeness. A final requirement was to test the completed application using the steps of the use cases and evaluating the results. Any area that did not produce the expected outcome was sent back to the developer. The developer reviewed the application to determine why it was not executing correctly, and modified the code to correct the problem. After applying the modification, the software was retested to confirm that all requirements were met. After all sections successfully completed the testing process, the application was ready to be placed into production.

3.2.4 Implementation Phase

After the product was thoroughly tested, the final stage was developing a way to make the application available to the end user. Visual Basic .NET includes a project template that can be used to create an installation routine. Because the product was developed in Visual Basic .NET, this template was used to create the deployment package. Compiling the installation project creates an .msi file that is used to deploy the application. To test the installation routine, the .msi file was executed on a non-production machine. After completing the install, the computer was reviewed to verify that the application was installed correctly. Once the setup routine was verified, the application was ready for the production environment.

3.3 Project Procedures and Formats

Before beginning the project, choices were made concerning the software applications that would be used during its lifetime. Several applications were selected to help manage and document the project. These included Microsoft Visio, Microsoft Word, and Microsoft Project.

Microsoft Visio was used extensively during the analysis stage of the project. The object-oriented design templates it provides eased the process of creating the use case, class, and class sequence diagrams. One difficulty experienced with using Visio was updating the sequence diagrams. It is problematic to insert a communication sequence into a sequence diagram because the object lifelines needed to be extended and communication sequences shifted down to complete the insertion.

Microsoft Word was used to consolidate the project's design documents. The combined document included the diagrams completed in Visio, the extended use case scenarios, and database requirements. This document provided the design requirements for the project.

Prior to beginning the project, the key tasks had to be identified. Microsoft Project was used to track the items that needed completed during each phase of the project. Because each task included a start and end date, this provided a schedule for the project.

3.4 Project Deliverables

Several items were developed throughout the different project phases. A list of these components was created prior to the start of the project. The next paragraphs will list several of the key artifacts and provide a brief description of their contribution to the project.

The administrative add-on developed during this project used a database to store authorized users and assessment item details. A database design document was created to provide the logical design for the project's data requirements. Components of the database design included the data dictionary, normal forms, and entity relationship diagram.

Because this project utilized an object-oriented approach, classes required by the application had to be defined. Class diagrams were used to identify the classes needed by the application and describe their relationships. They provided a roadmap for creating the needed objects and services.

Use cases were developed to identify the key tasks required of the application. Detailed steps were included in each use case to describe the success and alternate scenarios. Class sequence diagrams showed the communication required between the user and different application objects to fulfill each use case. The combination of the database design, class diagrams, and use case documents provided the requirements for the application.

A release candidate version of the software was an important deliverable of the project. The release candidate was a version of the application that incorporated the software requirements. It was tested to make sure that the requirements were met, and was ready to be installed in a production environment. To accomplish the software installation, a setup application was developed. The install program steps the user through the installation of the administrative tool, and ensures that the components required to run the software are in place.

3.5 Resource Requirements

Several computer applications were required to complete this project. For example, Visio was used to create the design artifacts, Project to identify and schedule project tasks, Word to combine the design artifacts, and Visual Basic .NET to create the application. Understanding how to use the software to implement the solution was an important prerequisite to successfully completing the project.

The author assumed multiple roles during the project. These roles included systems analyst, software developer, tester, and project manager. In the analysis phase, the system analysis duties involved creating the design artifacts for the project. Using these documents as a guide, the software application was created during the design phase of the project. Throughout the application's development, the software was tested to ensure that requirements were

being implemented. Examples of project management functions were developing the project plan and updating it as tasks were completed.

3.5 Outcomes

Modeling the application requirements was a crucial initial step in the project. Because Visio was used, these artifacts were available in an electronic format. This helped simplify the process of making modifications to the drawings.

Using the Visual Basic .NET development environment provided several benefits. First, it helped speed the process of creating the user interface because it provided interface elements that could be placed on the form using the mouse. The development environment's IntelliSense capability provided a list of properties and methods available for a particular object. Finally, the code could be stepped through line by line to follow the logic and identify problems that needed corrected.

3.6 Summary

Determining the phases that would be used and the tasks associated with each phase provided the framework for the project. The analysis, design, testing, and implementation phases covered all aspects of the project. Artifacts developed in prior phases were used to help produce the output of subsequent ones. A good illustration is how the

use cases, class diagrams, and sequence diagrams developed during the analysis phase were used to build the application during the development phase.

The project required the use of several software applications. An example is the use of Microsoft Project to build a plan for completing the project. The tasks included in the plan provided a checklist of activities that needed to be completed and a way to compare its actual completed date with the planned one.

Chapter 4

4.1 Project Origination

A project needed to be identified that would use Visual Basic .NET to illustrate application design patterns and provide examples for classroom discussion. Developing the add-on satisfied both of these requirements. A discussion of how the project met these needs is included in the next two paragraphs.

The add-on provides the test administrator with a convenient way to modify the assessment's content. To maintain the content, the add-on implemented three application design patterns. Examples of how the patterns were implemented are included in section 4.8.

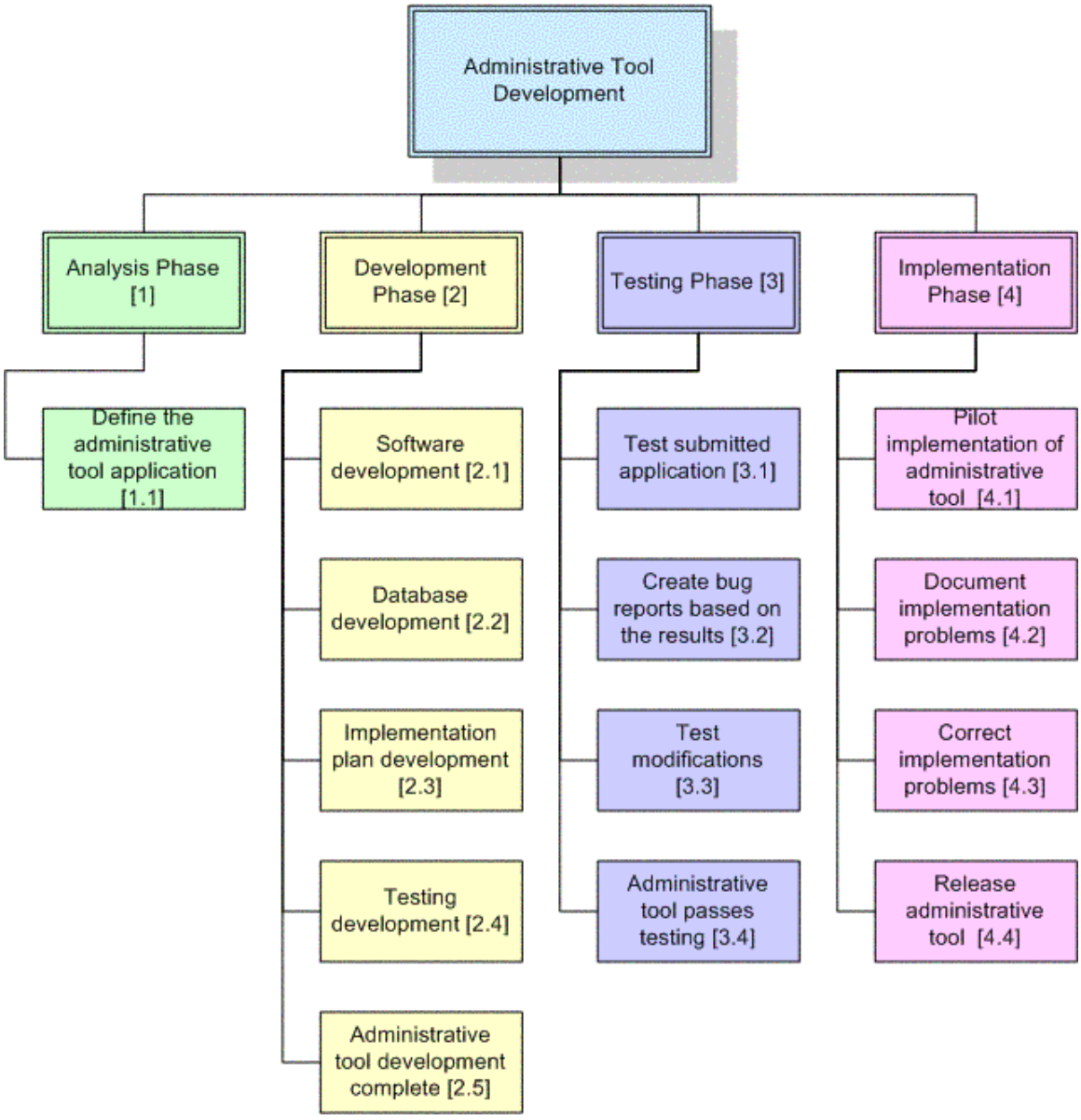
There are several teaching opportunities in the project. The class diagram can be reviewed with the source code to demonstrate how they are related. Following the use case scenarios while executing the software will illustrate how these documents affect the application's design. Reviewing the source code with the sequence diagrams will demonstrate how the diagrams were implemented in Visual Basic .NET.

4.2 Project Management Method

Microsoft Project was used to help manage this project. At the beginning of the project, a list of tasks that needed to be completed during each phase was developed. These tasks

were entered in Microsoft Project with planned start and end dates. After the task lists was reviewed for completeness, a baseline version of the project was saved. During the project, multiple tasks could be worked on simultaneously. Individual tasks were updated based on the percentage that was complete. Throughout the project, the work completed on the tasks was compared to the baseline to determine how the project was progressing. The WBS developed for the project is shown in Figure 1.

Figure 1 – WBS

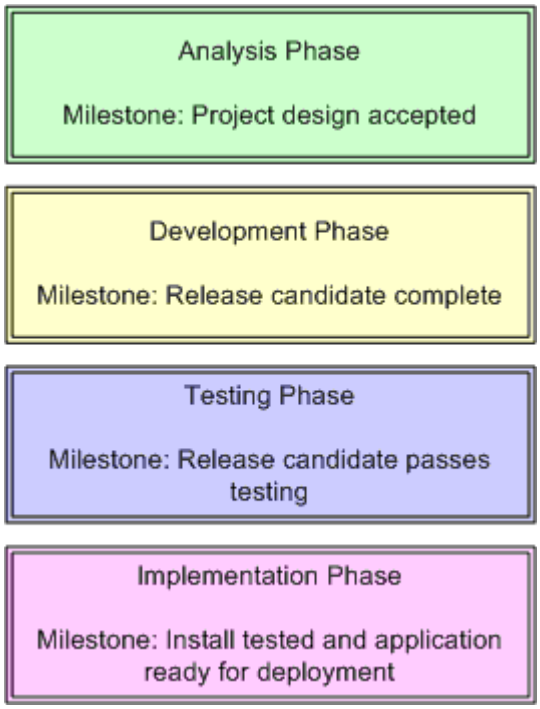


4.3 Project Milestones

Each phase defined for the project had important tasks whose completion signaled a significant advance in the project's development. However, not all tasks had to be completed in one phase before tasks in another phase were started. Frequently throughout the project, items in

multiple phases were being worked on at the same time. The milestones for each phase are listed in Figure 2.

Figure 2 – Project Milestones



Developing the use cases, class diagrams, sequence diagrams, and logical database design were key tasks of the analysis phase. After these items were created, they were reviewed for completeness. The milestone for this phase was reached when each design artifact had passed the review process.

The design phase focused on developing the application. This included using Visual Basic .NET to create the required code, and building the physical database from the logical design. At the end of this phase, a software package was prepared for testing.

Testing occurred in conjunction with the design phase. As units of the application were built, the code was reviewed to identify errors. Any errors that were found were corrected by updating the source code. Then the code was retested to ensure that the change corrected the problem and did not introduce new logic errors. The milestone for this phase was realized when the release candidate version of the software had passed all of the tests and was ready for implementation.

During the implementation phase, a setup project was created to install the application. The setup project was tested to verify that it installed the administrative tool correctly. The conclusion of this phase was making the setup project available for the production environment.

4.4 Project Plan Changes

The baseline plan that was built in Microsoft Project was different from the actual progression of the project. There were several reasons that this occurred. In practice, some tasks scheduled to begin later in the project were started earlier. The testing tasks provide a good example. These tasks were originally scheduled to begin after the development of the add-on was underway. However, testing actually took place throughout the entire coding process. This helped confirm the completeness of each software component as it was developed. In future projects, testing

tasks will be scheduled to coincide with the beginning of the development activities.

There were other tasks that took longer to finish than the time assigned to them in the baseline, requiring that subsequent tasks be adjusted. For instance, the time allocated to complete the release candidate took longer to complete due to additional workload assignments. Figure 3 shows a snapshot of the schedule comparing the baseline start and end dates to the actual ones.

Figure 3 – Schedule Snapshot

	Task Name	Start	Finish	% Work Complete	Baseline Start	Baseline Finish	Start Var.	Finish Var.
12	**Development Phase**	**Tue 2/7/06**	**Wed 5/3/06**	**100%**	**Tue 2/7/06**	**Tue 3/28/06**	**0 days**	**26.5 days**
13	**Software development**	**Tue 2/7/06**	**Mon 4/17/06**	**100%**	**Tue 2/7/06**	**Tue 3/28/06**	**0 days**	**14.5 days**
14	Identify software components	Tue 2/7/06	Tue 2/14/06	100%	Tue 2/7/06	Tue 2/14/06	0 days	0 days
15	Develop software components	Tue 2/14/06	Mon 4/17/06	100%	Tue 2/14/06	Tue 3/28/06	0 days	14.5 days

Microsoft Project was used to identify the percentage completed of each scheduled tasks. This report helped identify which tasks were slipping in the schedule. The working start and end dates for individual tasks were updated throughout the project's lifetime. When a task was finished, it was marked as 100% complete.

Microsoft Project worked well for scheduling tasks and reporting work completed. Reporting work completed for this project was accomplished by using the status date. Microsoft Project can automatically modify the start and end dates based on the status date, but the project manager needs to configure these settings. The settings are available by selecting Tools, Options, and clicking the Calculation tab.

4.5 Project Goals

Several goals were accomplished through this project. The add-on developed during the project allows the test's administrator to update the content and scores for the online assessment. This functionality was confirmed by using the add-on to retrieve current test content, update it, and save the changes. The technology skills assessment application was then opened to verify that the content changes made with the add-on were displayed correctly.

Research was completed to identify application templates that could be implemented in the project. Design methods that improve a Visual Basic .NET application's performance were investigated and included in the software. Through its design documents and application templates, the project will provide multiple examples for future projects and classroom instruction.

4.6 Project Review

The project provided a good opportunity to implement a variety of techniques taught in the object-oriented analysis and design courses offered at Regis University. Development of the analysis artifacts were based on discussions that took place during these courses. These documents were important to the success of the project because they provided the foundation for its development.

During past software development projects, adjustments to the schedule and resources were required to ensure the

project's successful completion. This project was no exception. A key to keeping the project on track was to identify the changes that were needed as soon as possible. There are usually more options available to adjust a schedule when the issue is identified in a timely manner. Two of the issues encountered in this project and how they were mitigated are included in the following paragraphs.

As the application was developed, a need for changes to some of the class attributes defined in the class diagram was identified. Because this change was discovered early in the development cycle, it was implemented without requiring extensive code modifications. The modifications would have been more difficult to make if they had not be discovered until final product testing. To avoid changes to the class attributes in future projects, the review team will spend additional time comparing the sequence diagrams to the class diagrams to verify completeness.

As the project continued, the development phase took longer to complete than was originally scheduled. The primary factor for this was the reassignment of the software developer to other projects. Completion dates for tasks were modified to accommodate the change in personnel availability. Involving the project manager in personnel scheduling decisions is a way to keep this from having a negative effect on the project's success.

4.7 Project Variables

The variable that seemed to have the most impact on this project was personnel. Because of the author's multiple roles in the project, the tasks were dependent on his availability to complete them. This project was scheduled to begin early in January. At the same time, the author was assigned to the oversight team for a MIS implementation project. This increased workload made it necessary to reschedule some of the project's tasks. Using Microsoft Project to maintain the schedule simplified the rearrangement of these items.

Another variable was completing the research to identify which application design patterns were appropriate for this project. The research results indicated that the MVC, Table Module, and Record Set patterns should be included in the design. These patterns were determined based on their applicability to a Windows application.

Implementing the patterns in the code was a third project variable. Visual Basic .NET uses form classes to control the user interface. It is easy to incorporate business logic into the form classes, but the selected patterns required that this code be maintained in separate classes. Planning the structure for each of the domain layer classes was important to the project's success.

4.8 Project Findings/Analyses

One of the project's goals was to identify and implement application design patterns in a Visual Basic .NET application. To accomplish this task, patterns for the presentation, domain, and data source layer were identified. These patterns were the MVC, Table Module, and Record Set.

The MVC separates the user interface from the domain logic. To implement the MVC pattern in this project, the Windows forms were created using the .NET integrated development environment (IDE). This allowed the software developer to build the user interface by selecting the required form elements from the toolbox, adding them to the form, and modifying property settings to achieve the correct control size and positioning on the form. After creating the visual appearance, the developer added code to the form class that responds when the user interacts with different controls on the form. However, the domain logic for the application is contained outside of the form in additional classes. Isolating the classes responsible for presenting the user interface from the domain logic allows the developer to focus on the visual presentation of the information without worrying about how the data is stored or retrieved.

To implement the Table Module pattern, each table required by the application is defined by a class. For this project, classes were created for User, Content, Answer, and Score. Each class included property sets and gets for each

of the table's columns. These class properties were populated by calling class methods that accessed the database. This design step is important because it isolated the data access methods to these classes. Methods used to retrieve and update information in the table were also incorporated. A benefit of this approach is that the user interface is unaware of the data access technology being used because there is no data access logic included in the user interface classes. The code listing in Figure 4 shows the property assignments for the User class.

Figure 4 – User Properties Code Listing

```
 Public Property UserID() As String
    Get
        Return strUserID
    End Get
    Set(ByVal Value As String)
        strUserID = Value
    End Set
End Property

Public Property FirstName() As String
    Get
        Return strFirstName
    End Get
    Set(ByVal Value As String)
        strFirstName = Value
    End Set
End Property

Public Property LastName() As String
    Get
        Return strLastName
    End Get
    Set(ByVal Value As String)
        strLastName = Value
    End Set
End Property

Public Property Password() As String
    Get
        Return strPassword
```

Figure 4 – User Properties Code Listing (continued)

```
        End Get
        Set(ByVal Value As String)
            strPassword = Value
        End Set
    End Property
```

The Record Set pattern was realized by including a reference to ADO .NET in the application using the System.Data namespace. This allowed the software to return Datasets from the database. Each table in the Dataset was used to populate the property values in the domain layer. When the user changed a property value and chose to update the table, the row stored in the Dataset was updated to reflect this change. Then the data was permanently updated in the database table by extracting the changed rows from the Dataset.

Figure 5 shows the initialization of the DBOperations object. The DBOperations class provided the data retrieval and update capabilities required by the project. Because data operations were required by several classes in the project, its design was critical to the project's success. Using this class simplified testing because it was inherited by all classes in the project that needed access to the database. This ensured that data operations were consistent throughout the application.

Isolating data operations in a separate class provided additional benefits beyond reuse. For example, it is possible that the database used by the application could be

changed to another vendor. If this happens, instead of requiring wholesale changes to the application, the changes will be primarily isolated to the DBOperations class. This will help ease the transition to another database technology. If a procedural approach had been used, the data access logic could occur throughout the source code. This would make it difficult to locate and change all of the database access logic to use a different database vendor.

Another goal realized by the project was identifying teaching examples for software development classes. One example is the establishment of coding standards and their use in a project. A standard identified in chapter 2 was documenting user-defined methods by including a description of their function, the input parameters required, and the output they produced. The code listing in Figure 5 demonstrates this documentation method. It also shows how the User object's attributes are updated with the values retrieved from the database.

Figure 5 – GetUser Code Listing

```
Public Function GetUser(ByVal UserID As String,
                        ByRef ErrorMsg As String) As User

    'PURPOSE: Retrieves the specified User from the database
    '         and populates the user object
    '
    'INPUT: UserID that specifies the ToolUser row to return,
    '       ByRef string used to return error message to the
    '       calling code
    '
    'OUTPUT: User object populated from the database
    '        ByRef string that returns error message to the
    '        calling code

    Dim myUser As New User
    Dim myDataReader As OleDb.OleDbDataReader
```

Figure 5 – GetUser Code Listing (continued)

```
        Try
            'Initialize DBOperation
            myUser.DBOperations(strCommonConnectionString)

            'Retrieve User from database
            myDataReader = myUser.generateReadOnlyRS
                           ("SELECT * FROM ToolUser " &
                            "WHERE ToolUserID = '" & UserID & "'")

            'Set Properties
            If myDataReader.Read = True Then 'Record found

                'Set myUser values using the column number
                With myUser
                    .strUserID = UserID
                    .strFirstName = myDataReader.Item(1)
                    .strLastName = myDataReader.Item(2)
                    .strPassword = myDataReader.Item(3)
                End With

                'Set OriginalUser Values using the column number
                myUser.OriginalUser = New User
                With myUser.OriginalUser
                    .strUserID = UserID
                    .strFirstName = myDataReader.Item(1)
                    .strLastName = myDataReader.Item(2)
                    .strPassword = myDataReader.Item(3)
                End With
            End If
        Catch myEX As Exception
            ErrorMsg = ("The following error occurred while " &
                        "attempting to retrieve the User!" &
                       vbNewLine & myEX.Message)
        End Try

        Return myUser

    End Function
```

4.9 Summary

The project provided a way for several goals to be achieved. One outcome was to provide a way to update the online assessment's content. This was accomplished by identifying the required software components, developing them with Visual Basic .NET, and delivering an application that provided this functionality.

Another goal was to provide examples for the different processes that occur during a software development project. The design documents and source code will serve as illustrations of the steps required to build a software application. This will help others understand the relationship between the different phases of the project.

Finally, identifying the MVC, Table Module, and Record Set design patterns will promote their inclusion in future development projects. Section 4.8 includes a discussion of how these patterns were implemented in the project.

Chapter 5

5.1 Project Lessons Learned

Every software development project the author has participated in has presented a unique set of challenges. These challenges provide a learning opportunity that can be applied to future endeavors. Some important lessons provided by this project included the process of identifying project tasks and their prerequisites, and the difficulty of assuming multiple project roles.

Planning for the project began by identifying what phases would be included, and the tasks that needed completed during each phase. Work on previous development projects helped in developing the list of requirements and time constraints for this project. However, past experiences were focused on the development activities of the project. This project presented opportunities to explore additional activities during the testing and implementation phases.

In smaller organizations, an IT staff member often assumes multiple roles. To experience the activities of the entire project, the author took the responsibility for each phase of the project. This presented several challenges. One problem was keeping the project documents updated after the primary analysis was completed. This was difficult because the project's focus had shifted to coding the application. Another difficulty was balancing additional work-related assignments with the project's tasks. When the schedule was

originally designed, it was based on the author's workload at that time. However, multiple team responsibilities for a new MIS project were added to the author's schedule. This made it more challenging to complete the project.

Determining what patterns were needed for the presentation, domain, and data source layers in a Windows application was an important learning experience. The MVC, Table Module, and Record Set patterns were discovered during research for the project. The project provided an opportunity to implement these patterns in a Visual Basic .NET application.

5.2 How the Project Could Have Been Improved

Assuming different roles during the project presented several challenges. One problem discussed in section 5.1 was keeping the analysis artifacts updated during the application development process. Having another project member or group committed to maintaining these documents would help correct this issue. This change would require that a communication process be defined and followed to ensure that the documents were updated correctly. It would also be advantageous to include a team member whose only role is managing the project. There are many activities that need to be tracked, and the project manager's role has enough project responsibilities without including additional development tasks.

Another project improvement realized through this assignment was the importance of involving the project manager in the organization's personnel decisions. As mentioned in section 5.1, the author's workload was significantly increased during the project's lifetime. Adjustments were required to keep the project on track. This experience demonstrated the importance of consulting with the project manager when incorporating changes that will affect a team member's project availability. This not only includes situations where a team member has additional responsibilities outside of the project, but also any planned vacations or additional dates that he or she will be unavailable.

Research was required to determine which design patterns would be used in the project. It took time to review the design patterns and find the appropriate ones. In future Visual Basic .NET projects, this research will not be necessary because the patterns and their implementation have been identified.

An area that took some additional development effort was determining how the domain logic should be implemented in the application. Because implementing the Table Module pattern in Visual Basic .NET was a new experience, this section took extra time to develop. However, once the initial class was built, the basic structure of the code could be transferred to the next one. For example, each class that implemented the Table Module template included a

method similar to the GetUser function shown in Figure 5. Recognizing this similarity helped increase the efficiency of the coding processing.

5.3 How the Project Fulfilled Expectations

Working on the project was a rewarding experience. It focused on building the add-on to simplify the process of updating the online assessment's content. Additional benefits provided by the project were identifying patterns useful for Visual Basic .NET development, building teaching examples, and experiencing multiple project roles.

The add-on gives the test's administrator the ability to review and update the content for any test section. It accomplishes this by allowing the administrator to select the section to display from a dropdown list. The add-on has passed testing and is ready to be implemented.

When teaching IT courses, it is beneficial to have examples to demonstrate the concepts being discussed. The deliverables from this project will provide illustrations from all phases of a software development project. During spring quarter, the project was used to demonstrate the transition from concept to code. This was accomplished by first reviewing the use case and sequence diagram for the logon process. After the design discussion, the students were shown the Visual Basic .NET source code that fulfilled the use case requirements. In future classes, the patterns incorporated in the application will also be discussed.

The author's primary project expertise was in the analysis and design of the software application. There were other team members responsible for quality assurance and implementation. Assuming multiple project roles allowed the project to be viewed from different perspectives. This knowledge will be useful in working with team members on future development projects.

5.4 Next Stage of the Project

It is important to think about the project beyond the fulfillment of the requirements for this course. There are a couple of future activities for the project. One way the project will continue beyond this assignment is by using it to provide examples for software development courses. These examples will supplement discussions about designing and implementing software solutions.

In the future, the add-on could be enhanced to provide additional functionality. The focus for this project was to allow the content to be updated with minimal changes being made to the online assessment application. The continuation of this project would be reengineering the add-on to provide more administrative control. For example, the current application only allows existing test content areas to be updated. A future development phase would also allow content areas to be added or removed.

5.5 Summary

The completion of the project achieved several objectives. Functionality provided by the add-on allows the content displayed by the skills assessment to be updated easily by the administrator. By completing tasks from each phase of the project, the author gained insight into the different skills that each stage required. The design documents and code developed for the project will provide models to supplement textbook material for future software development classes.

Some essential lessons were learned while working on this project. The importance of having a group assigned to maintain project documents was realized from this project experience. Changes to the author's workload had a significant effect on the project's schedule. This demonstrated the importance of management communicating with the project manager regarding personnel assignment changes.

An opportunity exists for further enhancement of the functionality provided by this project. The add-on addresses updating existent content areas. Future enhancements would allow it to insert and delete the assessment's content.

References

Fowler, M. (2003). *Patterns of Enterprise Application Architecture*. Boston, MA: Pearson Education, Inc.

Gunderloy, M. (2003). *MCAD/MCSD Developing and Implementing Windows-based Applications with Visual Basic .NET and Visual Studio .NET*. Que Publishing.

MSF Process Model v. 3.1. (2002). Microsoft. Retrieved May 26,2006 from http://download.microsoft.com/download/2/3/f/23f13f70-8e46-4f44-97f6-7dfb45010859/MSF%20Process%20Model%20v.%203.1.pdf

Rob, P., & Coronel, C. (2004). *Database Systems: Design, Implementation, & Management, Sixth Edition*. Boston, MA: Course Technology.

Schach, S. (2004) *Object-Oriented and Classical Software Engineering, Sixth Edition*. New York: McGraw-Hill.