Enhanced Application Modeling Techniques for Web Applications

All trademarks are property of their respective owners

Doctoral Dissertation

Student Number: 111086

Name: Munib Butt

Dated: 21st May 2003

Submitted to: Dr. M.R. Mousighi

All trademarks are property of their respective owners

Dedicated to my parents, my wife and Honey & Danu

All trademarks are property of their respective owners

Table of Contents

All trademarks are property of their respective owners

All trademarks are property of their respective owners

All trademarks are property of their respective owners

All trademarks are property of their respective owners

Chapter 1: The Topic

All trademarks are property of their respective owners

Introduction

The concept of Application modeling is a basic and essential component to the design and development of any computer-based application. Application modeling ensures that an application is developed to meet standards that have been set for level of Performance, Maintainability, Scalability, Reliability and Security. Application modeling is not restricted to only large scale enterprise applications but its scope covers a range of all applications from single user simple spreadsheet applications to Large Scale Web Applications that are accessed by millions of users at the same time all over the World by using the World Wide Web.

Application modeling is mostly a subject that is totally ignored or not used to an extent that its benefits are realized and hence it is often overlooked after the first few weeks or totally confined to a very small subset of the whole application. This can have very drastic effects to the overall design and effectiveness of the application once it is put into production.

Different types of Applications have been developed over the passage of time starting off with the large-scale applications, which ran on large Main Frame computers to applications designed to run on dumb terminals systems. Later, a new trend was born with the arrival of the Personal Computer and a new era was seen in computing.

All trademarks are property of their respective owners

The Desktop Enterprise application has been a very common type of application, which covers the aspect of thick clients installed on various user terminals all communicating with a central database via some fast connection Link. These applications have been in use for many years now. However, the latest trend, which has been seen is the development of Web Applications. These are applications that are not restricted to a certain geographical area but are available to the whole world if needed. Of course, these applications can be restricted in access if required and only be run and used within an organization on an Intranet or be restricted to a certain number of organizations running on an extranet. But, whatever the case the scope and range of web applications is certainly much more than any other typical applications, which have been developed in the past.

As we have seen above, Web Applications are the most common applications in development these days and even after the fall of the “dot com” era their utility and demand keeps on growing. Due to these factors my dissertation will concentrate on this type of application.

As we have already discussed, Application modeling is crucial to the design and development of any application. But when we come to Web Applications, especially ones that will be running on the World Wide Web, it becomes even more crucial because the range of users can grow from a few hundred to millions in a very small space of time. Hence, modeling the application to meet the

All trademarks are property of their respective owners

required needs of Performance, Reliability, Scalability, Maintainability and Security become even more crucial. Another factor, which is as much important as the above five mentioned factors, is the Usability or User experience.

The first five factors mentioned above will not be of much use if the user of a particular application cannot easily find the information he or she is looking for or cannot easily complete the task he or she wants to perform. Hence, modeling an application from the point of view of these six factors, is very crucial to the success of any application, especially web based applications.

Throughout my dissertation I will be looking at the various factors of Application modeling with reference to Web Applications, mainly the web applications designed for world wide web users as these exhibit the most crucial requirements of an efficient modeling solution. Why I call this dissertation “Enhanced Application modeling techniques for Web Applications” is because I will move ahead of the standard and conventional modeling technique used today where only certain subsets of an application are clearly modeled. I will try to engulf the whole application and model it to meet the requirements, which are defined as the ultimate goal to be achieved by the end product.

All trademarks are property of their respective owners

Background of the Topic

Here we will look at two main concepts, which are covered in the dissertation. First is the UML modeling technique, which will be used throughout the dissertation as the technique used for modeling. The second is the n-tier or n-Layer architecture. As we will be looking at the advantages of using a modeling technique for all tiers or layers of a web application, it is imperative to understand what these tiers or layers are and how they work together to make a web based application.

The UML modeling technique

Many software projects fail due to processes, which are not defined properly or have not been put into place. The need to have a disciplined approach in software development projects is essential. And to achieve this good process is one of the reasons for the evolution of UML (Unified Modeling Language).

As the Object Oriented technology gained momentum and Object Oriented languages became the languages of choice in the 80's and 90's most programmers started to adopt them. At this point System designers began to feel the need for an object-oriented methodology or representation method.

All trademarks are property of their respective owners

UML is a very simple and generic notation type language, which, is made of model like elements that can be used to model the requirements for the design of a software application. UML is not limited to software development domains and can also be used in other areas like business engineering and other types of projects. A good model is needed to communicate the concept of the software application to members of the project team and UML also reduces the costs of repetition of training and tooling when working on different projects. UML can help to give a better understanding of the application to be developed, speed up the development process, improve the code quality, support changing business needs as the application progresses, scale to larger projects and integrate the application easily with older legacy systems.

We will look below at why UML is the standard approach to modeling solutions. We will look at Objects, Messages, Classes, Inheritance and Polymorphism. The terms, Object Oriented Analysis, Object Oriented Design and Object Oriented Programming will be explained in a manner that is easy to understand.

UML as defined before stands for Unified Modeling Language. It is a language, which is used for specifying, visualizing, constructing and then documenting the various components of a software application and also other non-software applications or systems. These components constitute of a collection of standardized visual components, diagrams, and various types of notations.

All trademarks are property of their respective owners

When we represent any system by using UML, we mostly construct some or all of the following diagrams, which give us a better understanding of the whole system.

- Use Case Diagram
- Class Diagram
- State Diagram
- Sequence Diagram
- Collaboration Diagram
- Detail Class Diagram
- Deployment Diagram

We now take a look at each of these diagrams briefly:

Use case diagram

Description:

This diagram is a generalized description of how a particular system or sub-system will be used. It provides an overview of the intended functionality of the system and also looks at the possible alternate paths that may be chosen by the user. Both a layman as well as a professional can understand it.

Use Case Example:

Use Case Name: User searches for a book at an on-line bookstore.

All trademarks are property of their respective owners

Description: This user case covers the scenario where a user visits a web site and searches for a particular book by using a subject name.

Pre-requisite: User visits the on-line bookstore.

Primary Path: User types the subject name in the search text box.

User clicks the search button.

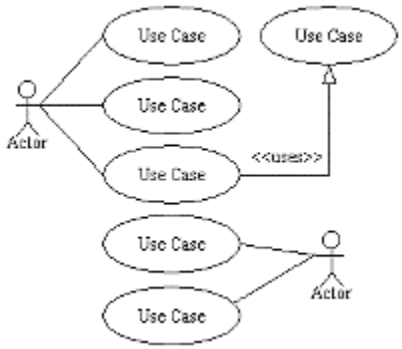
User is presented with a list of books that match his subject text.

User scrolls through the books.

User selects a book to purchase.

Alternate paths: User misspells the subject text.

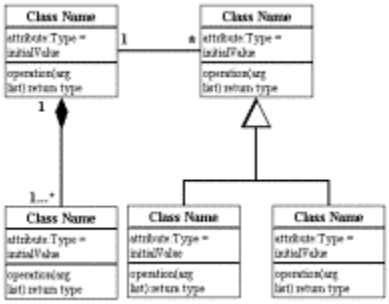
Storyboard: A diagram of how the screen would look like.



All trademarks are property of their respective owners

Class Diagram

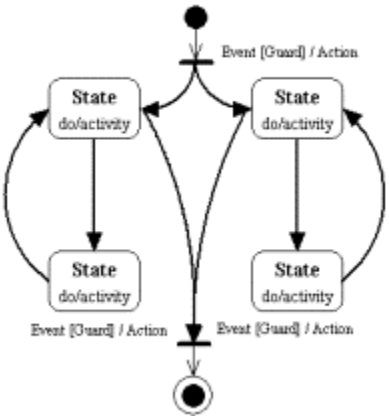
Class diagrams show the static structure of the various objects in the software application, their various properties and methods, and their relationships with other classes.



State diagram

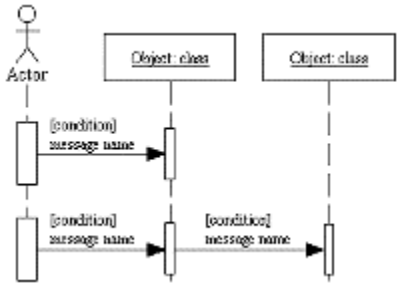
A state diagram shows the sequences of states an object goes through during its life cycle in response to different situations, along with its responses to these situations and actions.

All trademarks are property of their respective owners



Sequence Diagram

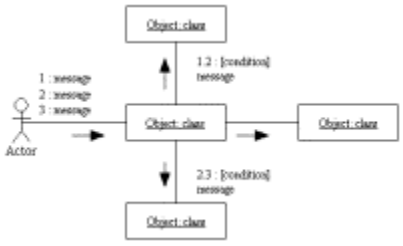
A sequence diagram and a collaboration diagram actually convey similar information but these two diagrams are expressed in different ways. A Sequence diagram shows the exact sequence of messages between objects, which is very suitable for modeling a real-time system. On the other hand a collaboration diagram shows the relationships between objects.



All trademarks are property of their respective owners

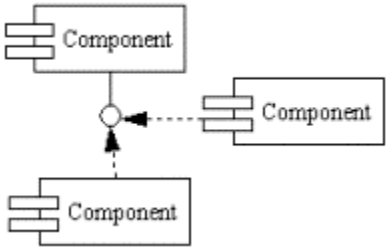
Collaboration diagram

As mentioned above the collaboration diagram shows the set of interactions between classes and the relationships among the objects.



Detail Class Diagram

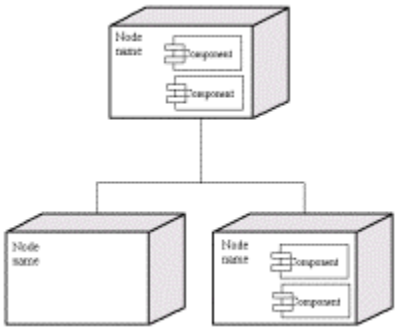
A detailed class diagram is done after the initial class diagram and most of the above diagrams have been completed. It consists of detailed information on the various classes that make up the entire software application.



All trademarks are property of their respective owners

Deployment Diagram

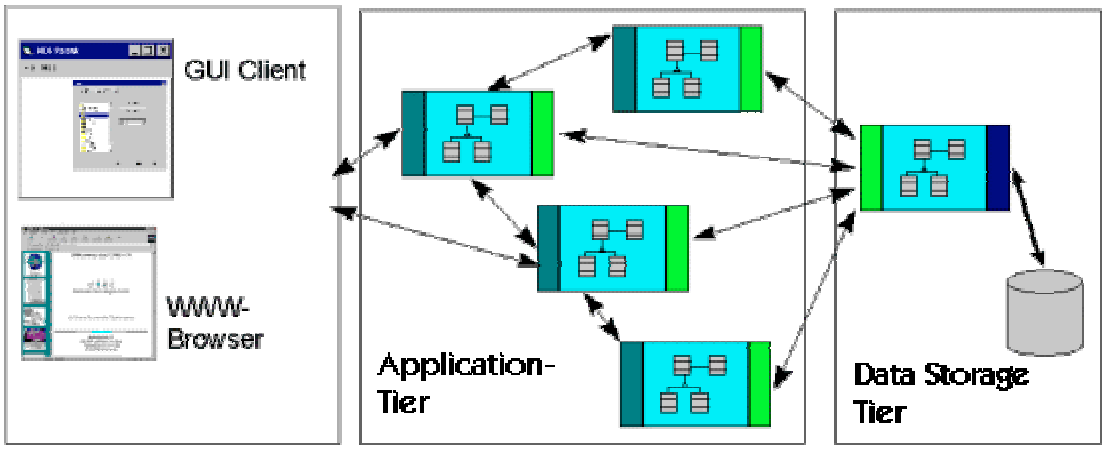
Although this diagram does not directly relate to the topic of the dissertation it is a very important diagram in the UML methodology. It details the architecture of the deployment of the software application.



We have taken a look at the UML modeling technique and this is the technique I will use in this dissertation to represent my idea of modeling the front end or Presentation Layer of a web application. I will be talking about the three tiers / layers or n-tiers of a web application, and how they relate to UML modeling. To explain the advantages of modeling all the tiers / layers, I would like to present a brief introduction to the 3-tier / layer or n-tier methodology before I move on and present the problem analysis and possible solution, which I plan to cover in this dissertation.

All trademarks are property of their respective owners

The n-tier software application architecture



We have to consider many elements when finalizing on the architecture of the application, such as performance, scalability, maintainability and future enhancement and extension issues. When we are deciding on which architecture is to be used, we have to take into consideration the above factors in relation to one another. For example, some choices that will increase the performance will have an impact on the scalability or future enhancements of your design, etc.

The first question that comes to mind is that what is n-tier or n-layer architecture? We can define “n-tier architecture” as being the architecture of an application which has at least three "logical" layers” or logical parts that are separate. Each layer interacts with only the layer directly below, and each layer has a specific function that it is responsible for.

All trademarks are property of their respective owners

Now this question comes to mind is that why should we use n-tier architecture for building our applications? The reason for this is that each layer can be physically located on different servers by making very small code changes, hence they can scale out and handle more load and hence make the overall application more scalable. In addition to this, each layer is a black box from other layers and this makes it possible to maintain one layer without having to worry about the other layers.

The feature of n-tier applications mentioned above is a very useful and powerful feature of n-tier applications, as new features or utilities can be added or previously developed ones can be enhanced, changed or deleted to a layer without redeploying the whole application. For example, by separating data access layer code from the business logic layer, we do not build in the data access methodology or components into the business logic layer. Hence, when the database server's change or a new data access methodology is required; we only need to change the data access layer code. Because business logic code stays the same, the business logic code does not need to be changed or recompiled and redeployed.

An n-tier application usually consists of three tiers, and they are known as the presentation tier / layer or front end as it is commonly known as, the business tier / layer and the data tier / layer. Let's take a look at what each tier or layer is responsible for.

All trademarks are property of their respective owners

Presentation Tier / Layer

The “Presentation Layer” is the layer that is responsible for displaying the user interface and creating this interface by communicating with the business tier objects. It is this layer, which provides the user experience and is a very crucial and important part of any web application. As seen in the diagram above, the Presentation layer can be a full-fledged application running on a user’s desktop for a desktop oriented application or it can be a web browser, which displays a web application.

Business Tier / Layer

The “Business Tier” is the layer, which is responsible for accessing the data tier to retrieve, update and delete data to and from the data tier and send the results back to the presentation tier. This layer is also responsible for receiving and processing the data sent from the presentation layer before it is sent to the data tier for storage.

Often the business layer is divided into two sub layers: the Business Logic Layer (BLL), and the Data Access Layer (DAL). The Business Logic Layers are above the Data Access Layers. Hence the BLL uses DAL objects. DAL is responsible

All trademarks are property of their respective owners

for accessing data and forwarding it to BLL. The business-oriented logic is coded in the BLL. On the other hand the DAL is just a generic interface to the underlying Data Tier.

Data Tier / Layer

The “Data tier” is the database or the data source itself. This can either be a Relational Database Server like MS-SQL Server or Oracle or a lighter desktop type database like MS-Access, however it's not limited to just these relational databases. It could also be an XML or a simple flat file, which contains the data. However, throughout this dissertation I will use MS-SQL Server, which is a relational database server as the Data tier as this is one of the most common databases used with web applications.

Now that we have covered the basics of what UML modeling is and what a n-tier architecture application is too, we will move on to the next step in the next topic and see what problem situation arises when this UML modeling technique is applied to a subset of the n-tier architecture application.

All trademarks are property of their respective owners

Statement of the problem situation

In the previous topic we have taken a look at what the UML modeling technique is all about and what it attempts to achieve. We have also taken a brief look at what the n-tier (also known as 3-tier) architecture attempts to cover. Now, in this topic I would like to cover what problems are faced when we try to use this UML modeling technique (or any other modeling technique for that matter) on only a subset of the whole application architecture.

I have worked in the software engineering field for over 14 years now and have been involved in the design of various types of application architectures. In addition to this field experience, I have also read quite a few numbers of books and articles related to application architecture modeling with special reference to web applications. The main downsize I see in all these implementations of UML modeling is the scope on which this very useful technique is applied.

In almost all cases the UML modeling technique or other application modeling techniques used were limited to the design of middle-tier classes and objects and also to the design of the database structure. This was a very useful step and provided a very solid foundation to the startup of the application. Although this procedure was also not very strictly followed after the initial phase of the application development process, it did provide a good foundation to start from. I will not ponder why this modeling process is not strictly followed into the

All trademarks are property of their respective owners

development and later phases, as this is not the topic I want to emphasize in my dissertation.

Now, with a solid design of the database structure and a number of various classes derived from the various use case diagrams we have a very good foundation to start our application from. However, what I saw in a number of applications especially web applications was that the user interface or user experience is a very crucial factor. The above modeling of the business and data tier was not enough to provide a successful application.

The common method used to express the Presentation Layer was a series of storyboards, which can also be called conceptual designs of what the front end or Presentation Layer, would possibly look like. These would mostly be attained from other sites with similar functionalities. Although, these would give the design team a good idea of what the Presentation Layer or Interface should look like, it was not enough in most cases.

The result was that in most cases screens and front end components would be developed that were not acceptable to the application users. Things would be missing, not placed properly, flow of events would be out of sync with each other and the whole effort in the modeling of the below business or middle tier and the data tier would be of no use.

All trademarks are property of their respective owners

The above would result in redoing the Presentation Layer a number of times and ultimately this would take the project out of the timeline and budget that was allocated to it.

My recommendation, which I will try to prove in this dissertation, is that the Presentation Layer components must also be modeled with the same amount of effort and detail as the Business Layer and Data Layer components. We must know where and what is going to appear on which display screen and report and why is it being placed there. In this way we would easily be able to identify pieces that are missing from the Presentation Layer and improve the user experience. Another benefit of modeling the Presentation Layer is that in most cases many different classes are defined in the Business Layer, which are never reflected in the Presentation Layer as the two have never been modeled side by side.

My dissertation of “Enhanced Application Modeling techniques for Web Applications” tries to look at the benefits that we can get out of modeling a web application at all three tiers and most emphasis will be given to the modeling of the Presentation Layer and matching it with the Business Tier as the modeling of the Business Tier and Data Tier are processes that are mostly followed at this time. We will try to see things that are looked after which might have been missed if the modeling technique was not used for the Presentation Layer.

All trademarks are property of their respective owners

This dissertation will try to provide a good practice that can be used when we are modeling web application architectures. It will not provide any magical tool or product, which can model the Presentation layer and match its various components with the Business or Middle Tier. This is a design methodology which when used can provide a better design of the application. The modeling technique used will be the same UML modeling technique and the application architecture will be the same n-tier architecture. I do not plan to redefine these two standards. I just plan to provide a better practice or technique to apply these technologies and get a better application architecture for a web application.

All trademarks are property of their respective owners

Purpose of the study

The purpose of the study of this dissertation is to see if we gain any benefits by adding a modeling phase to the Presentation Layer. Normally Web and other Applications have the modeling phase applied to only the Business and Data Layer of the application.

Now when we talk about benefits of adding a modeling Phase to the Presentation Layer of a Web Application, a number of things are covered. These include the following:

- Improved Application Architecture
- Improved understanding of the whole application
- Improved User Experience
- Faster Development Time
- Faster Application Finalization Time
- Code Efficiency
- Reduction of bugs
- Reduction of Architecture flaws
- Improved flow of events from Presentation Layer to the Business or Middle Layer
- Improved Application Integration
- Easier Quality Assurance of the Application

All trademarks are property of their respective owners

We will look at both the positive and negative aspects on this theory although the focus of the study would be to prove that by adding the phase of “Modeling the Presentation Layer” we would gain the benefits that have been mentioned above. We will not be trying to prove otherwise but these points will also be covered at various points in the study.

All trademarks are property of their respective owners

Objectives to be investigated

The objectives that are to be investigated in this study is that when we model a Web Application by applying the rules of UML to all the Layers in the application, we gain a number of benefits. This is compared to the old style used by most architects and development teams where only the Middle or Business Layer and Data Layer were modeled. Here other simple techniques like storyboards etc. were used to help design and develop the Presentation Layer.

Now, the main question that comes to mind is what are those benefits that we are looking for. A small list of these benefits is listed in the previous topic. Proving that we achieve these benefits by applying the modeling technique to the Presentation Layer of a web application is the objective of this study. If we achieve each separate benefit or not will cover one objective of this study. We will also see the negative side of each of these benefits as sometimes to gain a certain feature like Scalability or Maintainability we might sacrifice more important things like required Performance Levels etc.

The complete list of each objective to be covered in this study can be found in the “Statement of Hypothesis” topic.

All trademarks are property of their respective owners

Conceptual or substantive assumptions

The conceptual assumption that we will be making during the study of this dissertation is as below:

When we apply the UML modeling techniques and model our Application by covering all Layers (Presentation, Middle or Business and Data) then we have an application architecture, which is:

- More complete
- Robust
- Less prone to errors
- Less prone to missed features
- Easy to understand
- Easy to design
- Easy to develop
- Easy to test

When we look at the old style of UML modeling of application architectures, where only the Middle or business tier and the data tier were modeled, we see that we have a lot of gain by applying the modeling technique to the Presentation Layer.

All trademarks are property of their respective owners

However, when we talk of also modeling the Presentation layer and mapping it to the Business Layer, which in turn is mapped to the Data Layer, this does include an extra effort. Hence, extra time and effort has to be put into this process. Now, does the gain by using this modeling of the Presentation Layer justify that we put in this extra effort? This is something we will look in this study.

However, the assumption made will be that we do gain the above factors when we model the Presentation Layer in addition to the Middle Layer and the Data Layer and this effort is worth the time and effort. Hence we will be proving this and not otherwise. However, when we prove that there is a gain in the above mentioned factors by applying this technique we will also prove that leaving this process out has many negative effects.

Another very important assumption that is being made is that a UML or another modeling technique is being used to design the Application Architecture. However, we will be using UML as the modeling technique in this study.

There are certainly many applications especially web applications, which are designed without any modeling technique. We will not be looking at these applications as they clearly fall outside the scope of this study. Hence, we ignore these types of applications.

All trademarks are property of their respective owners

We assume the using a modeling technique is essential to the design of any application, which most people will agree with. The point we are to prove is that we can have an improved Application Architecture when we model the Presentation Layer in addition to the Middle and Data Layer, which is not normally done.

All trademarks are property of their respective owners

Rationale and theoretical framework

The theoretical framework that will be used in this study to prove that we gain in many areas as mentioned earlier by applying the UML modeling technique to the Presentation Layer as well as the Middle Layer and the Data Layer is detailed below:

We will model a simple web application, which consists of a few pages and will use this application architecture model to further design and develop the application. This web application will be modeled in two separate phases. One in which all Layers including the Presentation Layer will be modeled and the other in which the Presentation Layer will not be modeled. Then these two models will be designed and developed into a functional web application and during this phase and after completion, we will look at the differences we had by using these two application architecture models.

We will compare the two final products and the design, development and testing phase of them as well. Here we will see what advantages we gained by adding the Presentation layer to the modeling solution. This will complete the study of this dissertation.

The application that will be developed will consist of a few pages, which include the following:

All trademarks are property of their respective owners

- Main Page
- Products Categories Page
- Products List Page
- Product Detail Page
- Registration Page
- Login Screen

Using Microsoft .NET development methodology we will develop this application. It will consist of ASPX Pages on the Presentation Layer, VB.NET Components at the Middle or Business Layer and a Microsoft SQL Server Database at the Data Layer. Details of the Application Architecture using both models will be covered in Chapter 3.

All trademarks are property of their respective owners

Delineation of the research problem

The delineation of the research problem is that when we model the Presentation Layer of a web application in addition to the Middle or Business Layer and the Data Layer we gain many substantial benefits. These benefits have been outlined in detail in the next topic. In the research we are trying to prove that by applying the UML modeling techniques on the Presentation Layer we gain all these benefits. Also when we leave this phase out, we loose complete control of the Application Architecture and then many benefits, which could have been attained, are missed out. The reason for this is that, we did not spend a handful of resources (both time and manpower) to model the Presentation Layer in addition to the Business or Middle Layer and the Data Layer.

All trademarks are property of their respective owners

Statement of hypothesis

In this topic, we will look at the different hypothesis, which we have formulated and will try to prove through this study:

- *We will have an "**Improved Application Architecture**" when we apply the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  This means that we will have a better mapping between the components, which reside on these three different Layers.

- *We will have an "**Improved understanding of the whole application**" when we apply the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  Here we will see that the Architects, Developers, Quality Assurance teams and all other members of the application team will be able to see the bigger picture. They will know how the application exists as a whole as compared to just looking at the back end components and not really knowing the user Interface section.

All trademarks are property of their respective owners

- *We will have an "**Improved User Experience**" when we apply the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  This is one of the main things we want to prove in this study as in this way can make sure that there are no components of the Middle and Data Layer that have not been covered in the Presentation Layer. This problem is very common in older applications where tables and classes were created for a certain feature but this feature was never reflected on the Presentation Layer.

- *We will achieve a "**Faster Development Time**" when we apply the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  This is a by-product of some of the other benefits that will be gained. As all members of the Application Team will have a better understanding of the application, the Developers will also be a able to develop the application much faster as they will understand the flow from the Presentation Layer to the Business Layer and onto the Data Layer.

All trademarks are property of their respective owners

- *We will achieve a "**Faster Application Finalization Time**" when we apply the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  This is a point closely related to having a better Application Architecture. As compared to old projects where due to lack of models at the Presentation Layer level, applications tended to change a lot as different ideas floated at different times because nothing was really modeled. And when finally something was finalized and developed in many cases it had to be redone as it did not match the requirements of the components at the business and data layer and also the business requirements as a whole. In our study we will see that once we have a complete model at each Layer of the Application, we can easily see what changes are required and where they have to be done. Hence things can be finalized at the design stage as compared to changing things during the development stage.

- *We will achieve "**Code Efficiency**" when we apply the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  This is again due to the fact that the Developers and Architects will have a good understanding of the whole picture. They can see the best and optimized way to write efficient code as compared to the old style where

All trademarks are property of their respective owners

many code style changes had to be made on the fly and this resulted in inefficient code in the final product.

- *We will have a "**Reduction of bugs**" when we apply the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  This closely related to the previous objective. Because the Developers and Architects will have a better, clear and whole picture of the application model, there will certainly be less bugs. This is as compared to the old scenario where things are frequently changed and new ideas and methods introduced to meet steep deadlines due to the frequent changes.

- *We will have a "**Reduction of Architecture flaws**" when we apply the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  This is linked to the objective of a better Architecture design. Naturally when every Layer has been closely modeled and we have defined the link between various components at different Layers of the application, there are less chances for flaws.

All trademarks are property of their respective owners

- *We will have an "**Improved flow of events from Presentation Layer to the Business or Middle Layer and finally to the Data Layer** " when we apply the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  When we model each Layer of the Application, we will certainly define flow of events between components on all three layers of the application. Hence, the flow will be much better then when we just model the application for the middle and data layer. Although in the latter case, the flow between the middle and data layer will be modeled but we will be missing a very important component, which is the presentation layer in the flow of events.

- *We will have "**Improved Application Integration**" when we apply the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  This is again a point related to the Application Architecture. Naturally, the Application Integration will be much better if we also incorporate the Presentation Layer into the Design as compared to just modeling the Business and Data Layer.

All trademarks are property of their respective owners

- *We will have an "**Easier Quality Assurance of the Application**" when we apply the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  When the entire team including the QA members have a complete understanding of the application and the flow of events in the application, it will be much easier for them to perform Quality Assurance and other testing on the application.

All trademarks are property of their respective owners

Importance of the study

The importance of this study is that we will try to prove the numerous benefits that can be achieved from modeling the Presentation Layer in addition to the other two Layers. This is an accepted fact that modeling an application gives us a better understanding of the application and helps us in designing, developing, testing and deploying the application. However, this modeling technique is only used on the Business or Middle Layer and the Data Layer in most applications today.

By showing the importance of modeling the Presentation layer, this study will help many application development teams especially those engaged in design and development of web applications to improve in their application design and development. It will make the teams more productive and help them produce a better final product.

All trademarks are property of their respective owners

Chapter 2: Review of Related References

All trademarks are property of their respective owners

Organization of the present chapter

In the first section of the topic “Historical background” in this chapter, we will take a look at the “*History of the Unified Modeling Language”* (UML). We will see how this language came into being and how it has evolved over the time and become the language of choice for application architecture design.

In the second section of the topic “Historical background” in this chapter, we will take a brief look at the “*History of the 3-tier Architecture*”. We will look at the start and evolution of the n-tier or layer Application Architecture and how it has been accepted as an industry standard.

All trademarks are property of their respective owners

Historical background

History of UML

The identifiable object-oriented modeling languages began to appear between mid-1970 and the late 1980s. During this time various methodologists experimented with different approaches to object-oriented analysis and design. The number of identified modeling languages increased from less than 10 to more than 50 during the period between 1989-1994.

Most users of Object Oriented methods had trouble finding complete satisfaction in any one of these modeling languages, and hence adding to the "method wars." By the mid-1990s, new iterations of these methods began to appear and these methods began to incorporate each other's techniques, and a few clearly prominent methods began to emerge.

The development of UML (Unified Modeling Language) began in late 1994 when Grady Booch and Jim Rumbaugh of Rational Software Corporation began their work on unifying the Booch and OMT (Object Modeling Technique) methods. In the fall of 1995, Ivar Jacobson and his "Objectory" company also joined Rational and this unification effort. He added in the OOSE (Object-Oriented Software Engineering) method.

All trademarks are property of their respective owners

Being the primary authors of the Booch, OMT, and OOSE methods, Grady Booch, Jim Rumbaugh, and Ivar Jacobson were much motivated to create a Unified Modeling Language for the following three reasons.

- First, the methods designed by the three were already evolving and moving towards each other independently. Hence, It made more sense to continue that development together rather than apart, eliminating the potential for any unnecessary and voluntary differences that would further confuse the final users.

- Second, by unifying the semantics and notation of the modeling languages, they could bring some stability to the object-oriented marketplace, allowing projects to settle on one mature modeling technique language and allowing the companies building tools to focus on delivering more useful features.

- Third, they expected that their by collaboration among themselves they would gain improvements in all three earlier separate methods, helping them to see the lessons learned and to address problems that none of their methods previously handled well as an individual method.

All trademarks are property of their respective owners

The efforts of Booch, Rumbaugh, and Jacobson resulted in the release of the UML 0.9 and 0.91 documents in June and October of 1996. During 1996, the UML authors invited and received feedback from the user community. They incorporated this feedback, but it was clear that additional focused attention was still required to improve the language.

While Rational was bringing the Unified Modeling Language together, efforts were being made on achieving the wider goal of a modeling language that would serve as an industry standard.

In early 1995, Ivar Jacobson (then Chief Technology Officer of Objectory) and Richard Soley (then Chief Technology Officer of OMG) decided to push even harder to achieve a standardization in the methods marketplace.

In June 1995, an OMG-hosted meeting of all major methodologists (or their representatives) resulted in the first worldwide agreement to seek the methodology standards, under the stand of the OMG process.

During 1996, it became very clear that many organizations saw UML (Unified Modeling Language) as important to their business. A Request for Proposal (RFP) issued by the Object Management Group (OMG) provided the reactant for these organizations to join forces around producing a combined RFP response. Rational Software established the UML Partners group with several organizations

All trademarks are property of their respective owners

and was willing to allocate resources to work toward a strong UML 1.0 definition. Those contributing most to the UML 1.0 definition included: Digital Equipment Corp., HP, i-Logix, IntelliCorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational Software, TI, and Unisys. This collaboration produced UML 1.0, a modeling language that was well defined, expressive, powerful, and generally applicable. This was submitted to the OMG in January 1997 as an initial RFP response.

In January 1997 IBM, ObjecTime, Platinum Technology, Ptech, Taskon, Reich Technologies and Softeam also submitted some separate RFP responses to the OMG.

These companies joined the UML partners to contribute and add their ideas, and together all these partners produced the revised UML 1.1 standard. The focus of the UML 1.1 release was to improve the clarity of the UML 1.0 semantics and to add the contributions from the new partners. It was submitted to the OMG for their consideration and was adopted in the fall of 1997.

All trademarks are property of their respective owners

History of the 3-tier Architecture

Many years ago, most of the computing environments consisted of mainframe computers, which were hooked to dumb terminals that only did processing at the mainframe itself.

Over the years, personal computers began to replace these dumb terminals. However, the processing continued to be done on the mainframe. The much-improved power of personal computers were largely not utilized or used only on a single application level. With so much computing power not being used, many companies started thinking about splitting, some of the processing between the mainframe computer and the PC terminal. Client/server technology evolved out of this idea.

The Client/server technology refers to the method in which software components interact to form a system that can be used for multiple users. This technology is a architecture design that creates a system allowing the distribution of computation, analysis, and presentation between PCs and one or more larger and more powerful computers on a network. Each particular function of an application resides on the computer, which is the best to handle that particular function. It is not essential that the client and server must reside on the same machine. In practice implementations, it is quite common to place a server at one site in a local area network (LAN) and the clients at the other sites. The client, a PC or workstation, is the requesting machine and the server, a simple file server,

All trademarks are property of their respective owners

mini or mainframe, is the machine, which provides a response to these requests. Clients may be running on a number of different operating systems and networks and this does not affect the ability to send requests and receive responses.

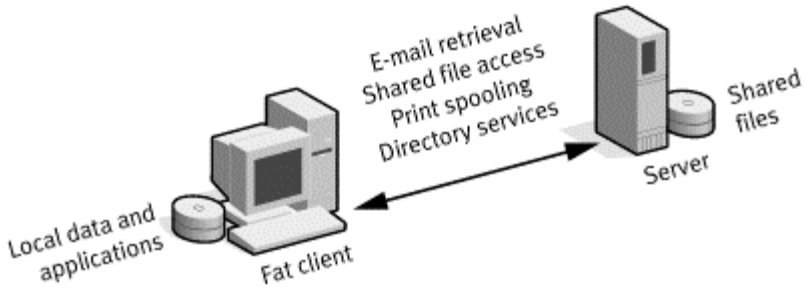
There has been an enormous growth in the client/server area. This growth seems to have come at by replacing the mainframe market. While the movement towards the client/server architecture from the mainframe architecture is gaining ground, there are many drawbacks since most of the client/server methodologies are have been completely put in place.

First generation systems are the 2-tiered architecture systems where a client component presents a graphical user interface (GUI) to the user, and responds to the user's actions to pass requests to a database server, which is running on a different machine.

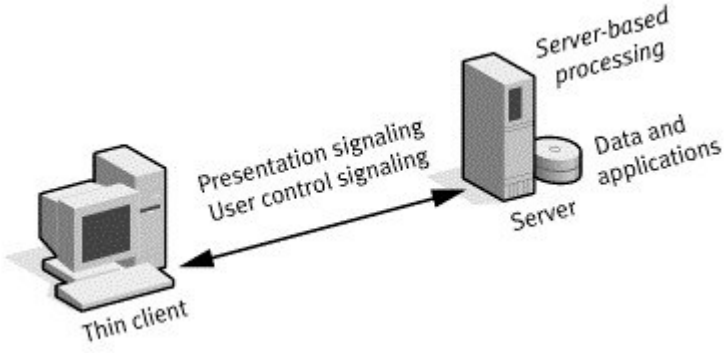
2-Tier Architectures

The client/server architecture applications started with a simple, 2-tiered model, which consisted of a client machine and an application server. The most common implementation of this is ‘fat client - thin’ server architecture, where the application logic in placed in the client. This is shown in the below figure. The database simply returns the results of queries and further processing is done on the client.

All trademarks are property of their respective owners



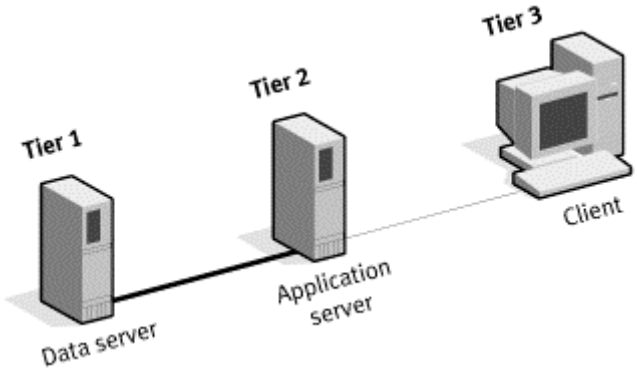
Another approach for the 2-tier Architecture Application is to use thin client - fat server, which runs procedures, which are stored on the database server. This is shown in the below figure. The term thin client mostly defines user devices, whose functionality has been minimized to allocate processing power to the stronger servers. However, in both cases exclusively the client handles the presentation. The processing is split between the client and server, and the data is stored on and accessed through the server.



All trademarks are property of their respective owners

3-Tier Architectures

We can further improve performance by using the 'N-tier' client-server architecture. By inserting a middle tier in between a client and server achieves a 3-tier configuration. The components of three-tiered architecture are divided into three layers: a presentation layer, application layer, and data layer, which must be logically separate from each other. The below diagram provides a graphical representation of this architecture. The 3-tier architecture attempts to overcome some of the limitations of 2-tier schemes by separating presentation, processing, and data into three separate and independent entities. The middle-tier components are usually coded in a portable, non-proprietary language such as C. However based on the operating system used, other more proprietary languages can be used as well. Middle-tier functionality servers may be developed as multithreaded components and can be accessed by multiple clients.



The client interacts with the middle tier via standard protocols such as API or RPC. The middle-tier interacts with the data tier via standard database protocols.

All trademarks are property of their respective owners

The middle-tier contains most of the application logic, translating client requests into database queries and other required actions, and translating data from the data tier into client required data. If the middle tier is located on the same host as the database, it can be tightly bound to the database via an embedded 3gl interface. However, the middle-tier can be separated on to another physical server to add more processing power.

N-Tier Architectures

The 3-tier architecture can be extended to N-tiers when the middle tier is extended to provide connections to various types of services. Dividing the application logic among various hosts can also create an N-tiered system.

As applications moved towards becoming Web-oriented, the Web server front ends can be used to distribute the networking required to service user requests, providing more scalability to the whole application. In this architecture as shown below, the client sends HTTP requests for information and displays the responses provided by the application system. On receiving the requests, the Web server either returns the content directly or passes it on to a specific application server, depending upon the type of request received.

All trademarks are property of their respective owners

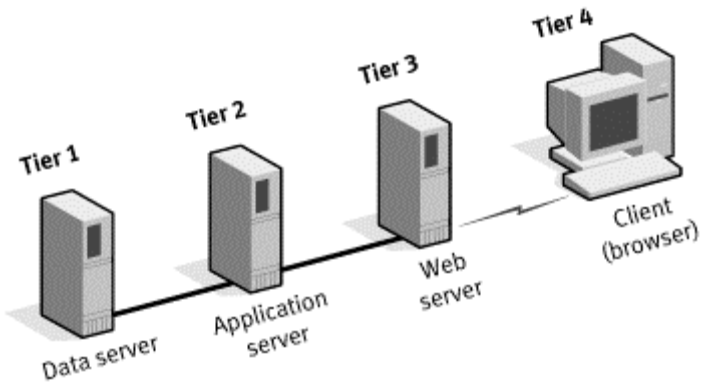


Figure 4. Web-Oriented N-Tiered Architecture

By separating each function, problems in the application can be more easily identified and removed by scaling the particular layer that is causing the problem. For example, if the Web server layer is the causing the application to run slowly, multiple Web servers can be deployed, with an appropriate server load-balancing solution to ensure the effective load balancing across the servers and the requests received.

All trademarks are property of their respective owners

Chapter 3: Methodology

All trademarks are property of their respective owners

Overview

In this chapter, we will cover the actual process where we try to prove that by modeling the Presentation Layer, we can improve the overall model of the web application being developed.

In the first topic (**Description of research methodology**), we will take a look at the web application, which will be designed by using both the modeling technique of 2-Layer modeling and then the enhanced technique of 3-layer modeling. In this topic we will see the various components of this application.

In the second topic (**Instrumentation**) we will take a look at the instrumentation. This is the application software that will be used to model these two above-mentioned designs. We will look at why we selected this software and what various types of diagrams will be used to represent the models in the 2-layer and 3-layer format.

In the third topic (**Selection of subject (concerning sample and population)**) we will discuss why we selected this particular application for this study to prove the benefits of modeling the Presentation Layer.

In the fourth topic (**Field, classroom, or laboratory procedures**) we will look at the various procedures we will follow to collect data, which will be analyzed during the study.

All trademarks are property of their respective owners

In the fifth topic (**Data collection and recording**) we will model the web application by covering the modeling technique on the middle or business layer and the data layer. Here we will first be modeling only 2 layers of the application. Then, we will model the same application, which has been modeled with a 2-layer approach but here we will also model the Presentation layer.

In the sixth topic (**Data processing and analysis (statistical analysis)**) we will take a look at the various steps that were faced by the developers during the development phase using the 3-layer model and the 2-layer model and the advantages that were gained using the 3-layer approach. We will also take a look at the end product, which has been developed after the development phase and we will see what advantages have been gained by using the 3-layer model as compared to the 2-layer model.

Finally, we will take a look at some methodology assumptions in the seventh topic. Finally, we will summarize the whole chapter before moving on to the next chapter where we list all our findings from the study.

All trademarks are property of their respective owners

Description of research methodology

In this topic we will take a look at the web application we want to develop. This is the web application, which we will be using as a basis for this study. We will then proceed and design the model of this application by applying the UML modeling techniques. We will first apply the UML modeling techniques only to the Business and Data Layer and then to the Presentation Layer as well.

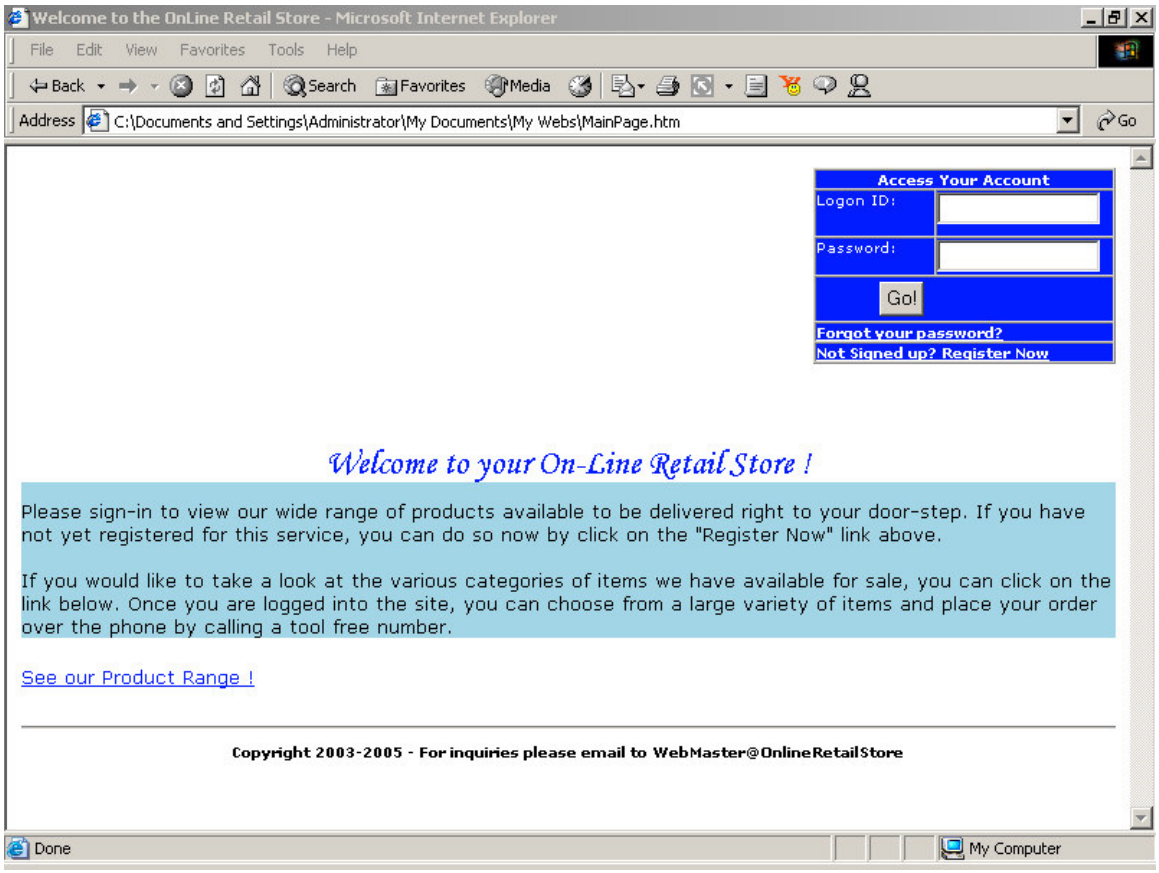
The Web Application

The web application will be designing in this study is an “Online Retail Application”. The scope of this web application is limited, as we want to demonstrate the advantages of modeling the Presentation Layer and not really create a fully functional web application for retail sales.

This web application starts with a user visiting the site by coming to the main page. Here the user has two options. The first is to register and receive an email with a username and password, to be used to enter the site. The second is to browse through the various categories of products that are available. However, if the user is not registered and has not signed in, he or she can only look at the various product categories and the category items. They cannot see an individual item and its details. Once a user has signed in, they can browse all the categories, product items and item pages. Once, they have made a selection of what to buy, the user can call a toll free number and place their order. They would then receive a confirmation by email.

All trademarks are property of their respective owners

The first page visited is that Main Page which will look as below:

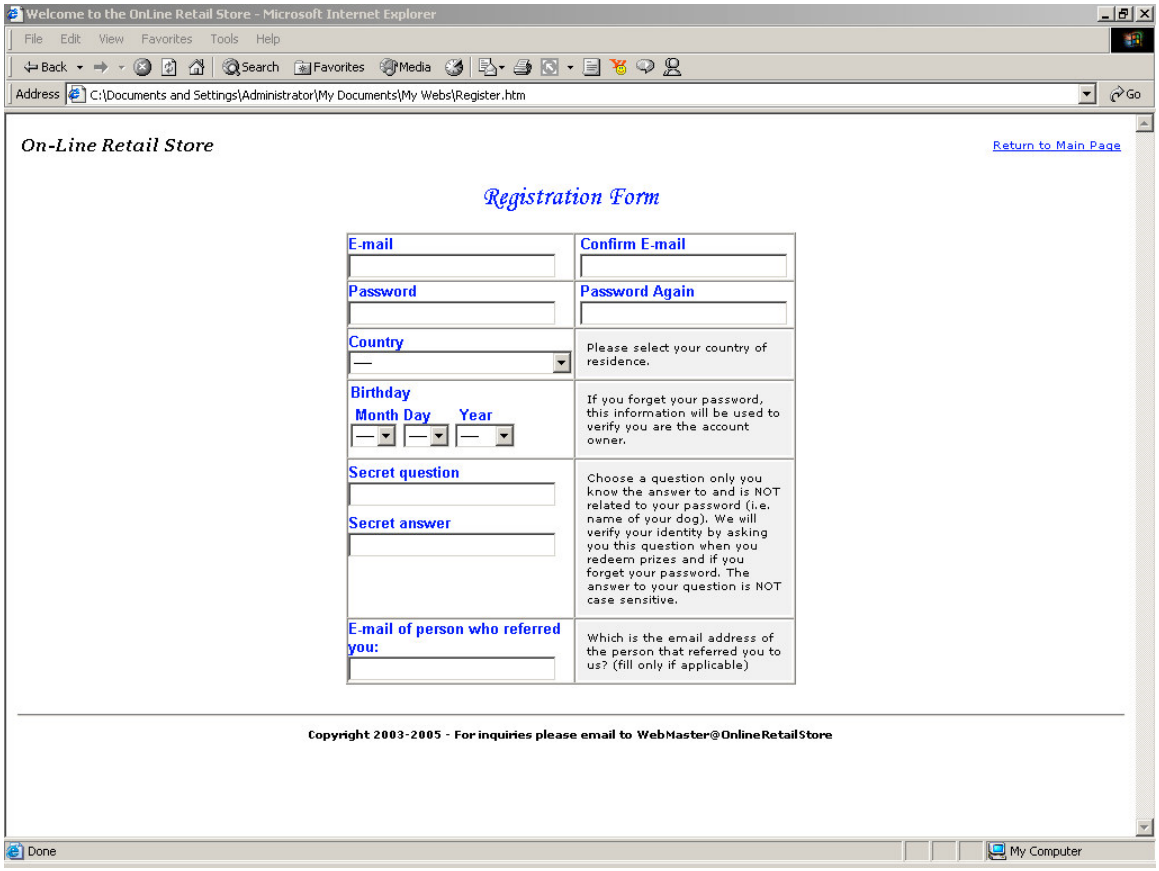


Here you can see that we have a small table at the top where a new user can register to use this web application. There is also an option for a user to retrieve their password, if he or she forgets their password. Then, there are the username and password text boxes for a user to sign-in. In the middle of the screen we see a welcome text message. And finally, at the bottom there is a link to the Product Categories. This is for a visitor to see the various product categories and category items. Of course, a visitor cannot see the detailed item page. And below

All trademarks are property of their respective owners

all is the page footer with some copyright information and contact information to the administrator of the web application.

Next, we take a look at the Registration screen.



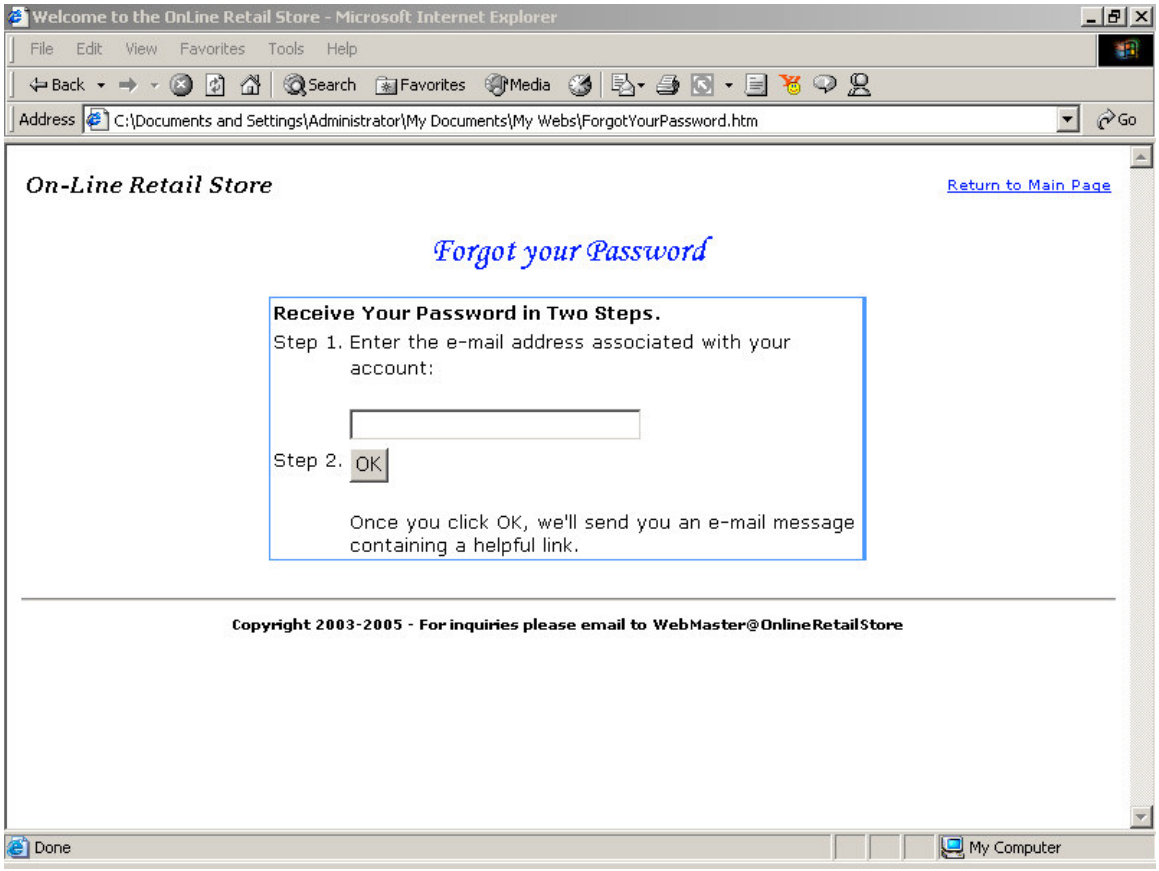
Here a user can enter all the information that is required to register and use the web site. These include:

1. Email
2. Password

All trademarks are property of their respective owners

3. Country
4. Birthday (for verification of lost password)
5. Secret Question
6. Secret Answer
7. Email of Person who referred you

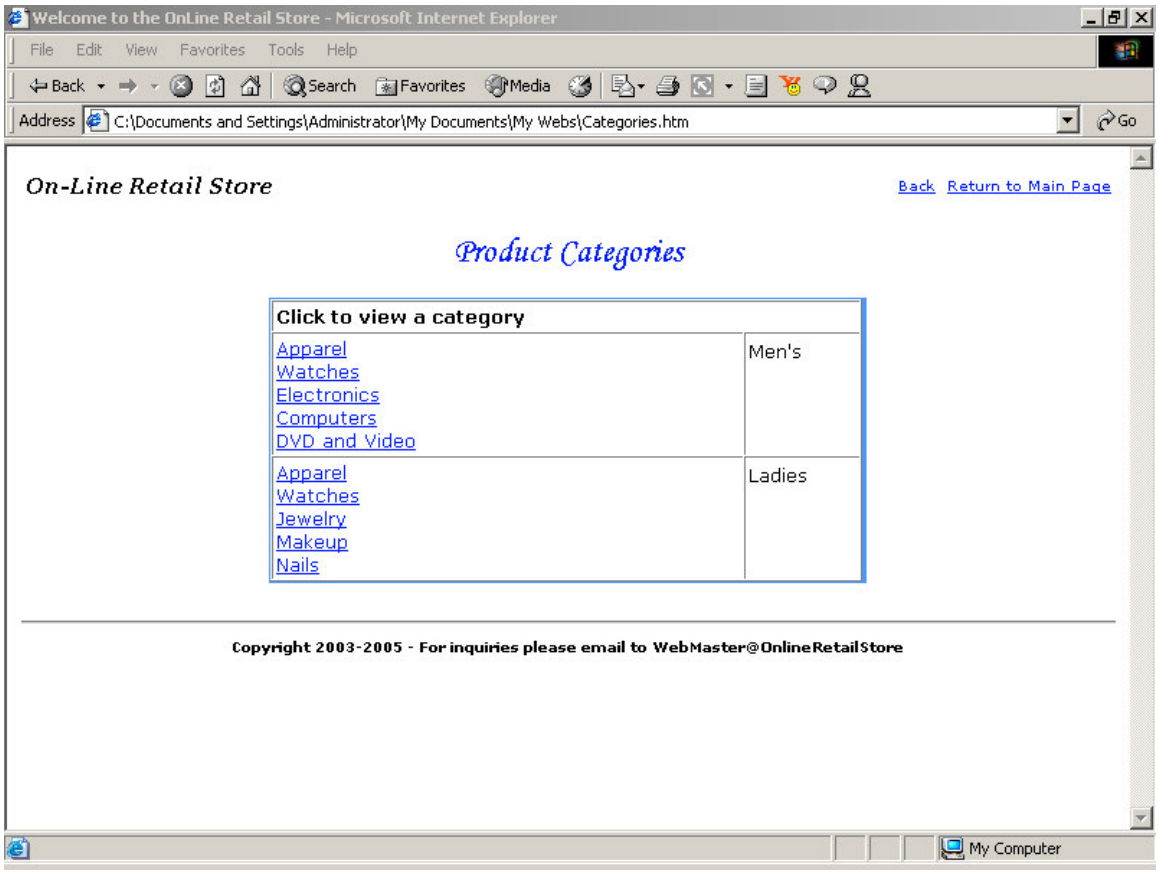
The fields 4,5 and 6 are required when a user loses his or her password. They will enter their email address on the Forgot your password screen as below:



All trademarks are property of their respective owners

Here, the user will enter his or her password and click OK. Then a user will receive an email asking him or her to either specify her or his birth date or secret question and answer. Once this information has been sent back to the Web Master, the Web Master will email the password to the user.

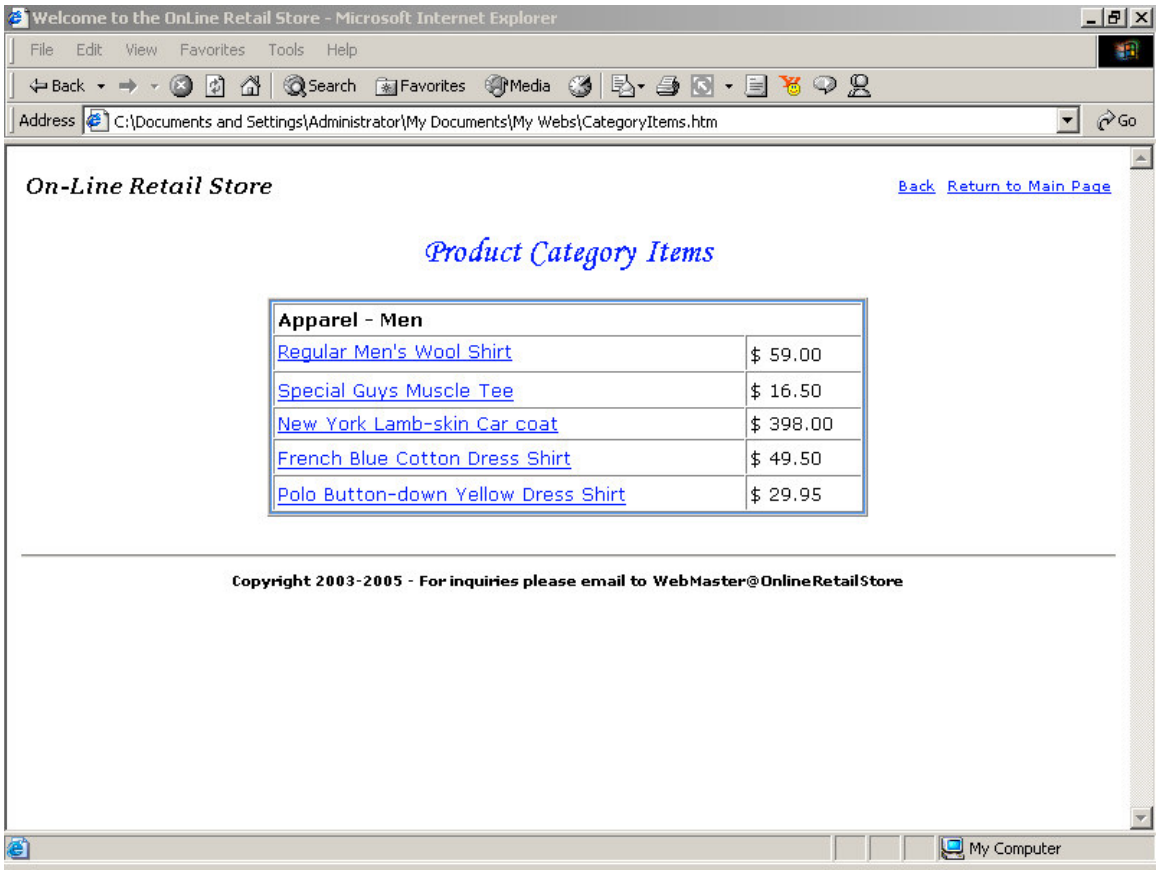
Next, we take a look at the Categories Main Page.



Here, we see that we have all our products grouped into various categories, which are either for men or for women. The user can see a list of these categories and then select the category he or she wants to browse and buy from.

All trademarks are property of their respective owners

Now, once the user has selected a category, he or she will see a complete range of products that fall under this particular category. The page will look as below:



On this page the user can see a list of all products that fall under the particular category he or she chose in the previous screen. The items are listed with the price at which they can be purchased. However, there are no details of any items on this page. And another change that can be seen is that on the top right hand corner there is a link to go back in addition to the link to return to the Main page. This has been done to provide the user a better navigation experience.

All trademarks are property of their respective owners

If the user is only a visitor who has not signed up on this site as yet and has come to the product category page by using the link on the main page, then the user will not be able to see the detailed item page but instead if the user tries to access the details, will be re-directed to the Main Page.

Now, let us look at the detailed product page.

Here we see that we have the complete description of the item along with the item number and the toll free phone number to call to place an order for this particular item. Once an order has been placed over the phone with a sales

All trademarks are property of their respective owners

representative, he or she will ship the goods at the address specified in the call using the billing information that has been given.

Now, that we have a good understanding of the web application we want to build, we will look at the various modeling techniques we will use to build this web application in the preceding topics.

All trademarks are property of their respective owners

Instrumentation

The Instrumentation tool we will be using to create our UML diagrams in this study will be Microsoft Visio 2002.

Now as we are interested in Microsoft Visio 2002's capabilities for software design and modeling, we have a choice of two versions: These are Visio 2002 Professional (Pro) or Visio for Enterprise Architects (EA).

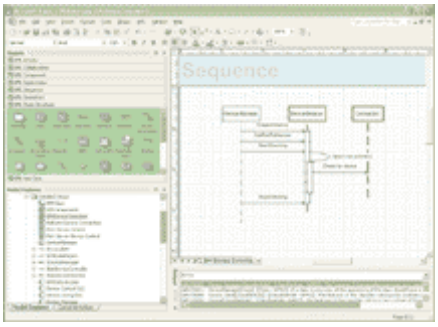
However, the EA version is only available as a bundled part of the Visual Studio .NET Enterprise Architect product. The EA version offers several useful features, including forward-engineering Visual Studio code from models, model semantic error checking, and custom reports, which are not available in the Professional version. So to really meet our design and modeling requirements, the best choice is to have Visual Studio .NET Enterprise Architect.

Microsoft Visio has a great amount of diagramming capabilities. We will just take a look at some here. Both the versions mentioned above can create a number of software and database design diagram types, including the full range of unified modeling language (UML) diagrams (which we will be creating in this study), component diagrams, data-flow diagrams and entity-relationship diagrams.

All trademarks are property of their respective owners

We can also create rich user interface (Presentation Layer) diagrams to model our graphical user interfaces (GUIs) visually.

Both versions can reverse-engineer Visual Studio projects back into UML models, or our databases into physical and logical models.



Microsoft Visio's integrated UML capabilities make it easy to select, add, and modify model elements and to create diagrams. We can add documentation with model elements. And the capability to mix pure UML diagrams with other free-form diagram elements is very useful when UML alone cannot convey the required ideas.

Also, the EA version offers semantic checking to keep us, with the rules of UML if required. It's easy to generate the code in the EA version and we can create custom templates for outputting our model documentation as code comments.

All trademarks are property of their respective owners

Selection of subject (concerning sample and population)

In this topic, we will take a look at why I have selected the Online Retail Application to present the idea of gaining substantial benefits by modeling the Presentation Layer in addition to the business or middle layer and the data layer in this study.

The Online Retail Application is a simple web application and although it has a limited number of actual pages, it covers most of the basic characteristics of a web application including User sign-in, registration, navigation, visual experience, categories, item links etc.

As the main aim of this study is to highlight the benefits of modeling the Presentation Layer, I have not selected a very large web application, as I do not want to indulge in the details of the web application itself and move away from the highlighting the benefits of a proper application model, which is the subject of this study.

However, a larger application might have shown the benefit of modeling the Presentation layer even more then this simple application, which I use here. But, all the important aspects will be clearly evident when we look at the benefits of modeling the Online Retail Application at each layer.

All trademarks are property of their respective owners

Field, classroom, or laboratory procedures

In this topic we will take a look at the procedures which we are going to use to collect the data, which will then be analyzed to confirm that we have gained a substantial number of benefits by modeling the web application at the Presentation Layer in addition to the Middle or Business layer and the Data Layer.

As the main benefit gained will be the better understanding of the web application as a whole, we will see this benefit in more efficient code, quick development time, less redoing of coding, testing and deployment. The analysts, developers, testers and the deployment team, will see all these benefits due to a better understanding of the application.

We will collect the required data using the following methods and analyze it partially in this chapter and then in more detail in the next chapter:

1. Interviews with various team members to see how well they understand the application and their level of satisfaction knowing the whole application picture on which they will be working.

All trademarks are property of their respective owners

2. Design, Development and Deployment time for a particular feature. This will be recorded on the development plan with actual and estimated time.

3. The time a particular feature was re-designed and then re-developed and finally re-deployed. This will be recorded on the development plan with actual and estimated time.

4. Missing application features in the Presentation Layer, which exist in either the Middle Layer or the Data Layer. These will be analyzed by the architects at various points of the development cycle and after the development has been completed.

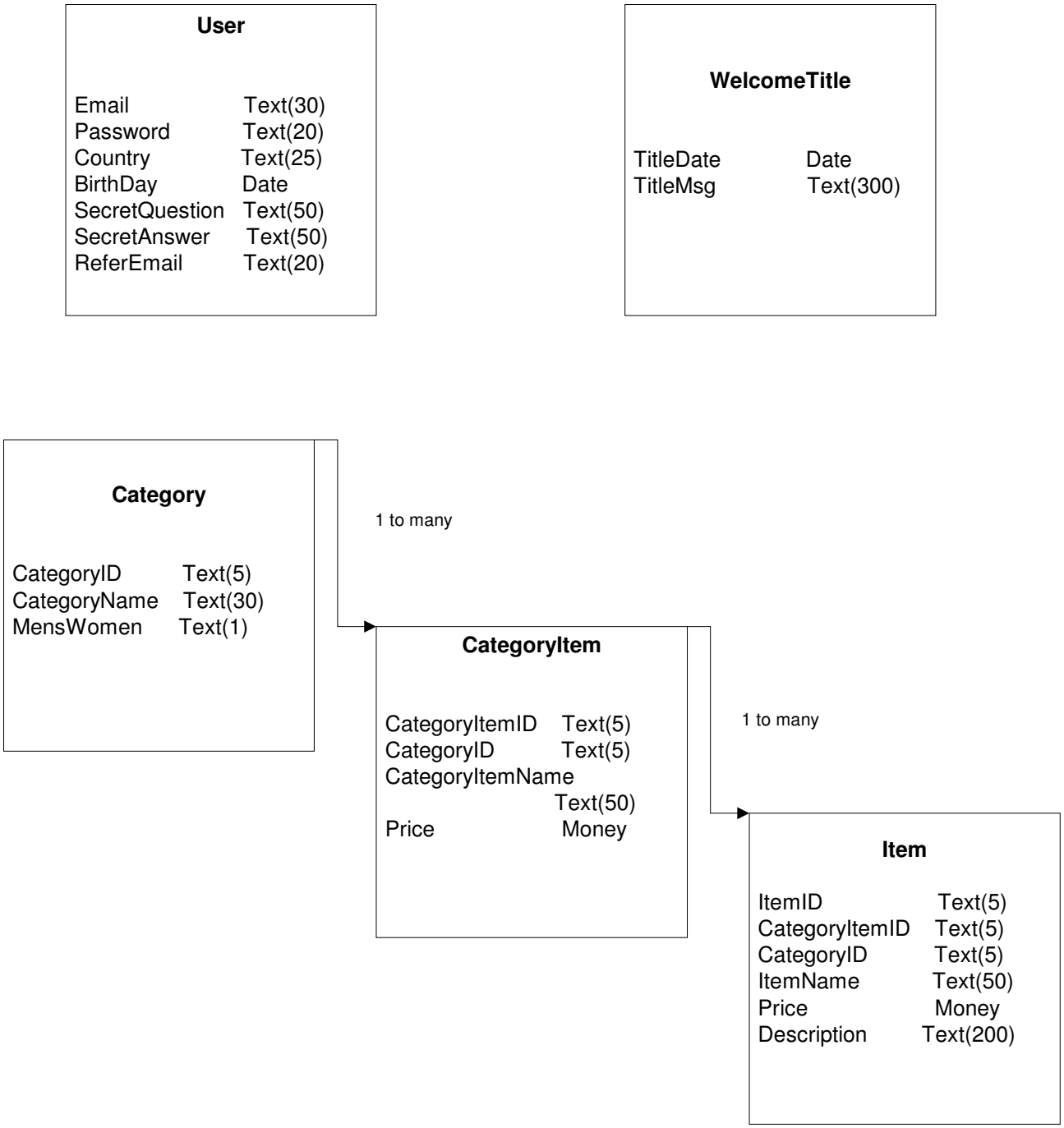
5. How easily can we add new features to the overall architecture of the application? This will also be analyzed during various phases of the application design, development and deployment and also after the final product is ready.

All trademarks are property of their respective owners

Data collection and recording

We will start the application design by first creating the data layer which will convert into respective tables in our Microsoft SQL Server 2000 Database. The Diagram for the data layer is below:

All trademarks are property of their respective owners



All trademarks are property of their respective owners

We now take a look at each entity and what it has been used for:

User Entity

This entity is used to store the user information in the database table. This entity is made up of the following fields, which will translate into columns:

Email

The unique ID of the user.

Password

The password the user will use to sign in to the web application.

Country

The country of the user.

Birthday

The birthday of the user. This information will be used to confirm the user information when sending out a password whenever requested through the "forgot password utility".

SecretQuestion

A Secret Question, which can be used as alternate to the birthday field.

All trademarks are property of their respective owners

Secret Answer

This is the answer for the Secret Question.

Refer Email

The email address of the person who referred the current user to this site. This could be used in the future for promotions etc.

WelcomeTitle Entity

This entity is used to store the information, which will be displayed on the main screen of the web application. This entity is made up of the following fields, which will translate into columns:

TitleDate

The Date on which we are to display the respective message.

TitleMsg

The Message, which will be displayed on the main screen.

All trademarks are property of their respective owners

Category Entity

This entity is used to store the information of a particular category of an item. This entity is made up of the following fields, which will translate into columns:

CategoryID

This is the unique Category ID for the category of the item.

CategoryName

This is the name of the category.

MensWomen

This is the attribute that defines if this category is for men or for women.

CategoryItem Entity

This entity is used to store the information of items that are related to a particular category. This entity is made up of the following fields, which will translate into columns:

CategoryItemID

This is the unique CategoryItem ID for the category item.

All trademarks are property of their respective owners

CategoryID

This is the unique Category ID. This is required to back track to the Main Category.

CategoryItemName

This is the Name of the CategoryItem.

Price

This is the price of a certain item.

Item Entity

This entity is used to store the information of a particular item. This entity is made up of the following fields, which will translate into columns:

ItemID

This is the unique Item ID for the particular item.

CategoryItemID

This is the CategoryItem ID for this particular item.

All trademarks are property of their respective owners

CategoryID

This is the Category ID for the particular item.

ItemName

This is the Item Name.

Description

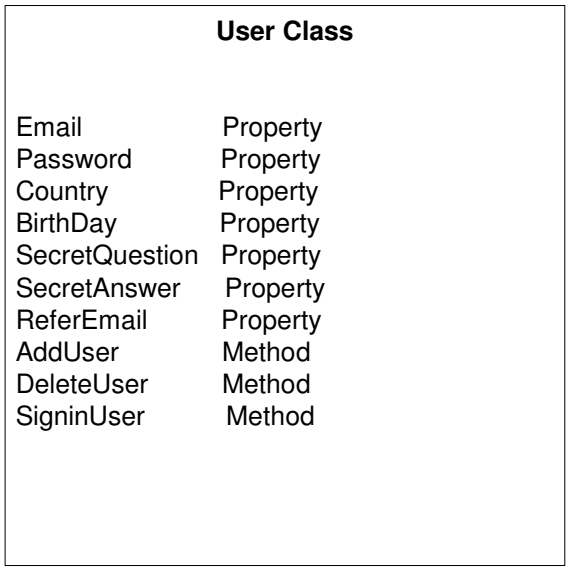
This is the detailed description of the Item.

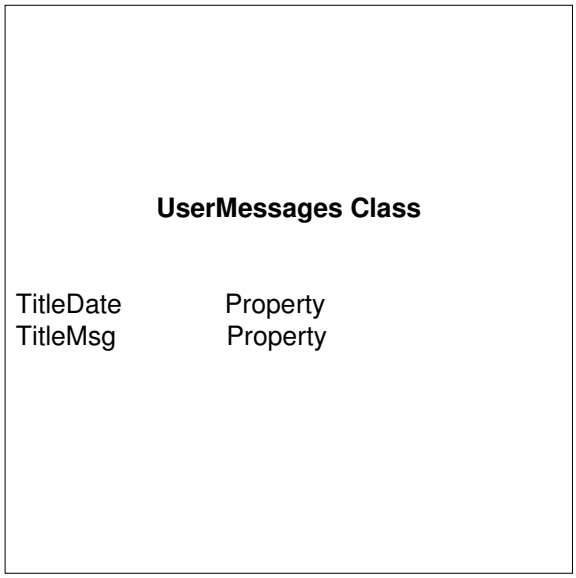
Price

This is the price of the item.

Next we will take a look at the business or middle Layer:

All trademarks are property of their respective owners





All trademarks are property of their respective owners

Category Class

CategoryID	Property
CategoryName	Property
MensWomen	Property
AddCategory	Method
DeleteCategory	Method

CategoryItem Class

CategoryItemID	Property
CategoryID	Property
CategoryItemName	Property
Price	Property
AddCategoryItem	Method
DeleteCategoryItem	Method

Item Class

ItemID	Property	DeleteItem	Method
CategoryItemID	Property	Description	Property
CategoryID	Property		
ItemName	Property		
Price	Property		
AddItem	Method		

All trademarks are property of their respective owners

Now, we will take a look at the different entities and the classes that are defined in each one of them.

User Entity

The User Entity will contain information for the user. This Entity has the following classes:

User Class

This Class contains information related to the registered user. This is the user who can use the site in full detail. He or she can browse through all the categories, the category items and the items in detail. The user can then call up a toll free number and provide all shipping and billing details and have the required items shipped.

The User class consists of the following properties (which define the attributes of the class) and methods (which define the behavior of the class):

Email

This property holds the unique ID of a particular user.

All trademarks are property of their respective owners

Password

The property holds the password, which will be used by the user to access the site.

Country

This property holds the country information for the user.

BirthDay

This property holds the birthday information of the user. This information is needed to verify the user's credentials in case the user forgets his or her password and requests that it is mailed to him or her.

SecretQuestion

This property is stored to be used as an alternate to the above property.

SecretAnswer

This property provides the answer, which is expected, from the user in response to the above question.

All trademarks are property of their respective owners

AddUser

This method is used to add a new user to the database table. It is however essential that all the above properties are provided before this method can be executed.

DeleteUser

This method is used to delete a user from the database table. This method required that the email property be provided before it can be called.

SigninUser

This method is used to confirm the combination of email and password and is called to verify a user and sign him or her into the web application.

All trademarks are property of their respective owners

Next, we take a look at the next entity:

User Messages Entity

The User Messages Entity will contain information, which will be displayed on the Main Screen. This Entity has the following classes:

UserMessages Class

The UserMessages Class is a class, which hold properties and methods for the various messages that are displayed on different screens in the web application. At this point we are only displaying one message on the Main screen and hence it hold information for only this one screen.

TitleDate

This property holds the Date on which the Message must be displayed.

TitleMsg

This property holds the Message, which must be displayed on the date above.

All trademarks are property of their respective owners

Next, we look at the final entity.

Items Entity

The Items entity contains all information that is related to items. It will contain information pertaining to the categories of the items, the various items, which fall under one category and the item details themselves. This entity has the following classes:

Category Class

The Category class is used to hold information, which is related to the category of an item. It consists of the following properties and methods:

CategoryID

This property holds the unique Category ID for the category of the item.

CategoryName

This property holds the name of the category.

MensWomen

This property holds the attribute that defines if this category is for men or for women.

All trademarks are property of their respective owners

AddCategory

This method adds a new category to the database table of categories. All the properties that have been defined above are required to call this method.

DeleteCategory

This method deletes a category for the database table of categories. This method required that the unique category ID field be populated.

The second class details are below:

CategoryItem Class

The CategoryItem class is used to hold information, which is related to the items, which fall under a certain category. It consists of the following properties and methods:

CategoryItemID

This property holds the unique CategoryItem ID for the category item.

CategoryID

This property holds the unique Category ID. This is required to back track to the Main Category.

All trademarks are property of their respective owners

CategoryItemName

This property holds the Name of the CategoryItem.

Price

This property holds the price of a certain item.

AddCategoryItem

This method adds a new category item to the respective database table.

DeleteCategoryItem

This method deletes a category from the database table.

The third class details are below:

Item Class

The Item class is used to hold information for a particular item. It consists of the following properties and methods:

ItemID

This property holds the unique Item ID for the particular item.

All trademarks are property of their respective owners

CategoryItemID

This property holds the CategoryItem ID for this particular item.

CategoryID

This property holds the Category ID for the particular item.

ItemName

This property holds the Item Name.

Description

This is the detailed description of the Item.

Price

This property holds the price of the item.

All trademarks are property of their respective owners

AddItem

This method adds an item to the database table. This requires that all properties mentioned above be populated before this method can be called.

DeleteItem

This method deletes an item from the database table and required that the ItemID field be populated.

Next, we will take a look at the Data Access Layer, which is a part of the business or middle-layer.

All trademarks are property of their respective owners

The Data Access Layer consists of a component, which has only one class, which is as below:

DataAccess Entity

DataAccess Class

This class is used for generic access to the particular database being used in the application. Hence, if we change our database or the data access technique we

All trademarks are property of their respective owners

do not have to change any part of our application part from this class. This class has the following methods:

AddRec

This method is used to add a record to a certain table in the database. The table name and the required field values are passed to this method as an array.

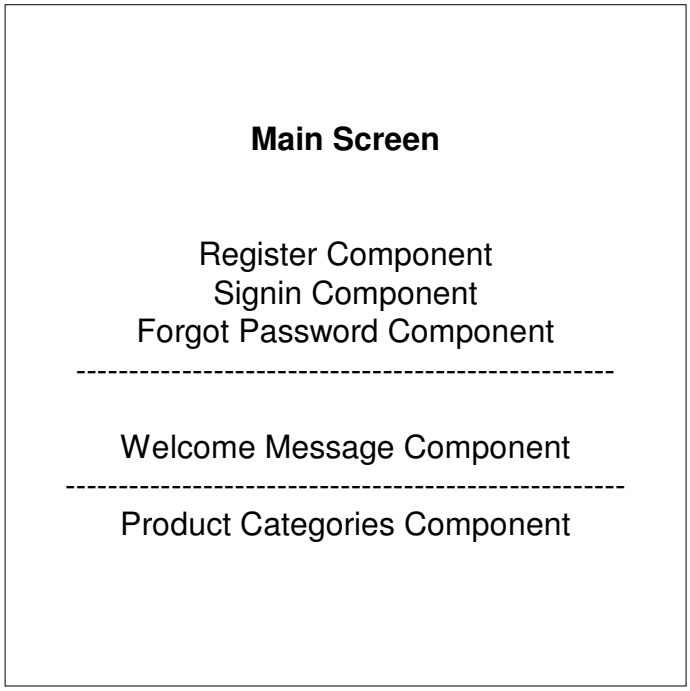
DeleteRec

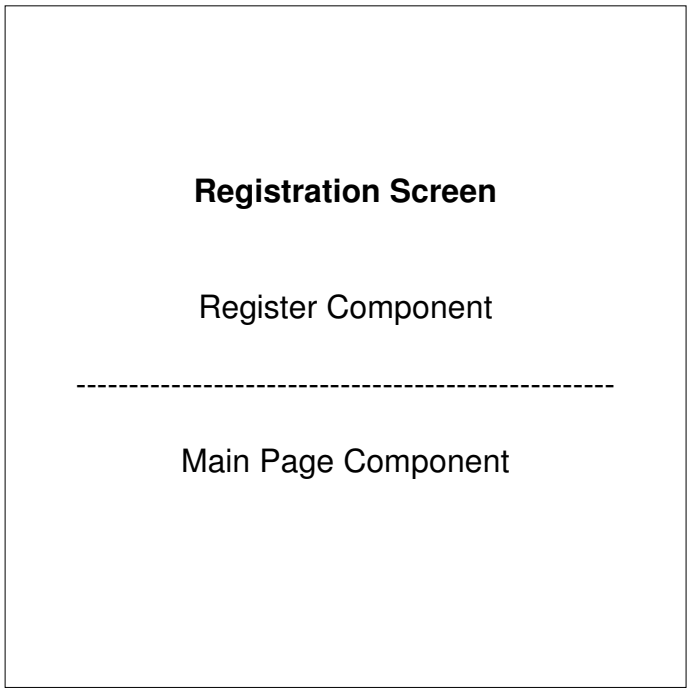
This method is used to delete a record from a certain table in the database. The table name and the required field values are passed to this method as an array.

Next, we will move on and model the Presentation Layer of the application.

All trademarks are property of their respective owners

Model of the Presentation Layer.





All trademarks are property of their respective owners

Forgot Password Screen

Forgot Password Component

Main Page Component

Category Screen

Category Component

Main Page Component
Navigation to Last Screen Component

All trademarks are property of their respective owners

CategoryItem Screen

CategoryItem Component

Main Page Component
Navigation to Last Screen Component

Item Screen

Item Component

Main Page Component
Navigation to Last Screen Component

All trademarks are property of their respective owners

Now we can look at the different screens in detail:

The Main Screen

The Main Screen is the first screen, which the user sees when he or she visits the web site. It has the following components on it:

Register Component

For a new user to register on the site.

Sign-in Component

For a registered user to sign-in and use the web site.

Forgot Password Component

For a user to receive his or her password if he or she have forgotten it.

Welcome Message Component

The Welcome Message that the user sees on the Main screen.

Product Categories Component

The Product Categories link for a non-registered user to view the product categories. However, a non-registered user cannot view item details.

All trademarks are property of their respective owners

The Registration Screen

The Registration Screen is the screen on which a user registers to use the complete web application. It has the following components on it:

Register Component

For a new user to register on the site.

Main Page Component

This is a link to return to the Main Page.

All trademarks are property of their respective owners

The Forgot Password Screen

The Forgot Password Screen is the screen on which a user can request for his or her email in case he or she forgets the password. It will be emailed to the user offline. It has the following components on it:

Forgot Password Component

For a user to obtain his or her forgotten password.

Main Page Component

This is a link to return to the Main Page.

All trademarks are property of their respective owners

The Category Screen

The Category Screen is the screen on which a user can view all categories of various items. These will be separate for men and women. It has the following components on it:

Category Component

This is the component, which displays the categories as required.

Main Page Component

This is a link to return to the Main Page.

Navigation to Last Screen Component

This is a link to the Last Page.

All trademarks are property of their respective owners

The CategoryItem Screen

The CategoryItem Screen is the screen on which a user can view all items related to a particular category. It has the following components on it:

CategoryItem Component

This is the component, which displays the category items as required.

Main Page Component

This is a link to return to the Main Page.

Navigation to Last Screen Component

This is a link to the Last Page.

All trademarks are property of their respective owners

The Item Screen

The Item Screen is the screen on which a user can view all details for a particular item. The user will also find information to order the item on this page. It has the following components on it:

Item Component

This is the component, which displays the item details as required.

Main Page Component

This is a link to return to the Main Page.

Navigation to Last Screen Component

This is a link to the Last Page.

In the next topic, we will look at the differences found by two teams when one of the teams used the model of three layers as compared to the other team, which used the old modeling technique where only two layers were used.

All trademarks are property of their respective owners

Data processing and analysis (statistical analysis)

We will take a look at the various observations and points that came across the two teams (one using a 3-layer model and the other using a 2-layer model) during the following three stages of the project:

- Pre-development
- Development
- Post-development

Pre-development

Some of the observations that were made in the Pre-development phase are as below:

1. Missing features

There is no component on the Presentation layer that allows for:

- adding new users
- deleting users
- adding categories
- deleting categories

All trademarks are property of their respective owners

- adding category items
- deleting category items
- adding items
- deleting items

This was a point, which came across due to the fact that the application had been modeled at all 3 layers. This was missed in the other implementation where the application was modeled for only 2 layers.

However, this point was noted and it was decide that in phase 2 of the project, an administration site would be created to perform all the features missing here. This is a very crucial thing that was missed in the 2-layer model.

2. Table and Component refinement

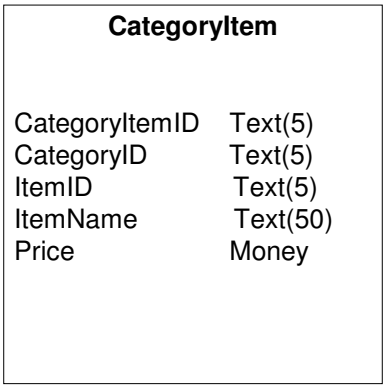
There is no need for a detailed “CategoryItem” table and component. The Category Item entity or table is just a joining table between the entities of item and category and must be refined.

This observation was also noted as a result of looking at the Presentation Layer model and was missed in the other implementation where there was a model for only 2 layers.

All trademarks are property of their respective owners

Hence the CategoryItem table and class were updated as shown in the below diagram:

All trademarks are property of their respective owners



Category Class

CategoryID	Property
CategoryName	Property
MensWomen	Property
AddCategory	Method
DeleteCategory	Method

CategoryItem Class

CategoryItemID	Property
CategoryID	Property
ItemID	Property
ItemName	Property
Price	Property
AddCategoryItem	Method
DeleteCategoryItem	Method

Item Class

ItemID	Property	DeleteItem	Method
CategoryItemID	Property	Description	Property
CategoryID	Property	AddItem	Method
ItemName	Property		
Price	Property		

All trademarks are property of their respective owners

The Presentation Layer component remains unchanged.

All trademarks are property of their respective owners

Development

Some of the observations that were made in the Pre-development phase are as below:

1. Dynamic Title generation

It was observed that the Titles must be generated dynamically rather then from a specified date, although the date could be maintained for purposes of backtracking. Hence, it was decided that this feature would also be added the Administration site to be developed in Phase 2. The team who used the 2-layer model did not note this point.

All trademarks are property of their respective owners

Post-development

Here we saw that the team, which used the 3-layer model, found and corrected the issues in the previous two steps. However, the other team, which, used a 2-layer model, found out of these problems at this stage when development was complete.

Hence, the team, which used a 2-layer approach, had to redo a lot of work and hence the time of development and finalization of the application was increased by a lot.

For the team, which worked with the 3-layer model, the process of design, development, testing and deployment went very smooth as all members of the team understood the process from start to end and had a clear picture of the whole application. This was clearly seen when various team members were asked about various aspects of the application.

The above was lacking in the team working on the application by using a 2-layer model. The various team members had a number of questions in their minds, which remained unanswered till the end and were not clear on many aspects of the application.

All trademarks are property of their respective owners

A few other things that were noted when comparing the team with a 2-layer model and the team with a 3-layer model during the whole cycle of the application development were as below:

Whole Picture seen in 3-layer model – Poor Story boards in 2-layer model

- "Improved Application Architecture"
- "Improved understanding of the whole application"
- "Faster Application Finalization Time"
- "Easier Quality Assurance of the Application"

Flow is different if we view category items on Main Page. We cannot beyond this without sign-in. easily visible in 3-layer model.

- "Improved flow of events from Presentation Layer to the Business or Middle Layer and finally to the Data Layer "
- "Reduction of Architecture flaws"

Message on main page is data driven. Point noted in development phase of 3-layer model.

- "Improved User Experience"
- "Improved Application Integration"

All trademarks are property of their respective owners

We can only get to the Item Page from Item List Page if signed in.

- “Faster Development Time”
- “Code Efficiency”
- “Reduction of bugs”

All trademarks are property of their respective owners

Methodology assumptions

The methodology assumptions made in this study is that the team working with a 2-layer model has constructed the business and data layer model as done above. However, they have not constructed the Presentation layer model as done above and used by the other team, which is working on the same application with a 3-layer model.

All points noted above are in comparison of the two teams. One with a detailed UML model of all three layers as prepared above and the second with a UML model of only the business and data layer. It is also assumed that the team with a 2-layer model did not have very comprehensive storyboards, which is mostly the case with teams working to develop web applications.

All trademarks are property of their respective owners

Summary

We have seen in this chapter that by preparing a detailed UML model for each layer we not only give each member of the team a good understanding of the entire application but also eliminate a lot of redundant work. We also trace and fix most architecture and development bugs much earlier as compared to the situation where we work with only a 2-layer UML model.

In the next chapter we will summarize all our findings and compare them one to one with our hypothesis, which we had set earlier.

All trademarks are property of their respective owners

Chapter 4: Analysis and Evaluation

All trademarks are property of their respective owners

Findings

The below findings were made in the study where we compared the development of a web application by two teams in which one was using a 2-layer UML model and the other was using a 3-layer UML model as a design archietcture to develop the application.

Description	3-Layer model	2-Layer Model
Clear understanding of the whole application	Yes	No
Solid Architecture	Yes	Partially
Easy to identify missing items on the Presentation Layer	Yes	No
Easy to change	Yes	No
Easy to develop	Yes	No. as details of the presentation layers are missing
Easy to test	Yes	No. as details of the presentation layers are missing
East to deploy	Yes	Yes

All trademarks are property of their respective owners

Easy to check for proper flow of features	Yes	Partially, as Presentation layer details are missing
Easy to update the architecture	Yes	No

All trademarks are property of their respective owners

Testing our hypothesis

Here, we will list all the hypothesis we have put down in this study and test each one of them against the observations we have made during the process of developing a web application using a 3-layer model.

- *We had an "**Improved Application Architecture**" when we applied the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  This was seen when we traced the missing tasks of adding users, deleting users etc. in the Presentation Layer.

- *We had an "**Improved understanding of the whole application**" when we applied the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  Here we saw that all members of the team which used the 3-layer model understood the whole application much better then the team which used a 2-layer model as they were not clear of the Presentation layer details.

All trademarks are property of their respective owners

- *We had an "**Improved User Experience**" when we applied the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  This was observed by the fact that we placed all necessary display components on the required pages and also that we traced that the main page message must be driven dynamically.

- *We achieved a "**Faster Development Time**" when we applied the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  This was due to the fact that the development team understood the application very well and that we identified most changes in the design and development phase as compared to the team, which worked with the 2-layer model and found architecture change requirements after development was completed.

- *We achieved a "**Faster Application Finalization Time**" when we applied the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

All trademarks are property of their respective owners

This was possible due to the fact that we detected and fixed most of the problems while we were in the design and development phase in the 3-layer model. However, for the team using the 2-layer model they found most architecture and other problems after development. Hence, in this way they had to come back and fix these issues, which increased the application finalization time.

- *We achieved "**Code Efficiency**" when we applied the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  This was due to the fact that the development team understood the application very well and that we identified most changes in the design and development phase as compared to the team, which worked with the 2-layer model and found architecture change requirements after development was completed.

- *We had a "**Reduction of bugs**" when we applied the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  This was due to the fact that the development team understood the application very well and that we identified most changes in the design and development phase as compared to the team, which worked with the

All trademarks are property of their respective owners

2-layer model and found architecture change requirements after development was completed.

- *We had a "**Reduction of Architecture flaws**" when we applied the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  This was possible because we modeled the whole application and we could see the entire picture rather then just one aspect of the application.

- *We had an "**Improved flow of events from Presentation Layer to the Business or Middle Layer and finally to the Data Layer** " when we applied the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  This was possible because we modeled the whole application and we could see the entire picture rather then just one aspect of the application.

- *We had "**Improved Application Integration**" when we applied the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

All trademarks are property of their respective owners

This was possible because we modeled the whole application and we could see the entire picture rather then just one aspect of the application architecture.

- *We had an "**Easier Quality Assurance of the Application**" when we applied the modeling technique to the Presentation Layer in addition to the Middle and Data Layer.*

  This was due to the fact that the QA team understood the application very well and that we identified most changes in the design and development phase as compared to the team, which worked with the 2-layer model and found architecture change requirements after development was completed. Hence, as changes were made earlier, the QA team could easier incorporate these changes in their test plans.

All trademarks are property of their respective owners

Factual information

All information that has been presented in the previous topic is based on facts either by figures updated in the project plans or feedback that was directly received from various team members. No assumptions have been made in the findings.

All trademarks are property of their respective owners

Evaluation

The evaluation that can be made from this study is that we gained numerous benefits by applying the UML modeling technique to the Presentation Layer in addition to the Middle or business layer and the data layer.

All trademarks are property of their respective owners

Summary of “Findings.”

Please see above.

All trademarks are property of their respective owners

Chapter 5: Summary, Conclusions and Recommendations

All trademarks are property of their respective owners

Summary

Looking back at the study we looked at the concept of applying the UML modeling technique to the Presentation Layer in addition to the Middle and Data layer.

In Chapter 1, we formulated our hypothesis that we will gain a substantial number of benefits by doing so. We set the following as our goals to achieve by modeling the Presentation Layer:

- Improved Application Architecture
- Improved understanding of the whole application
- Improved User Experience
- Faster Development Time
- Faster Application Finalization Time
- Code Efficiency
- Reduction of bugs
- Reduction of Architecture flaws
- Improved flow of events from Presentation Layer to the Business or Middle Layer and finally to the Data Layer
- Improved Application Integration
- Easier Quality Assurance of the Application

All trademarks are property of their respective owners

In Chapter 2, we looked back at the history of UML and how it has evolved into a language of choice to model our application architectures. We also look at the n-tier methodology in which we separate our application into logical layers to improve Performance, Scalability, Maintenance, Security and Enhancements.

In Chapter 3, we then started practical work to prove our hypothesis. Here we looked at a small web application project, which needed to be created. We modeled each layer of the application starting from the data layer and then moving to the business layer and finally the Presentation Layer.

We then handed two model architectures to two teams. One team got a model based on 3 –layers of the application to be developed and the second got a model of 2-layers.

We then observed how the design, development and finally post-development phases proceeded for both the teams and the changes and enhancements they had to make and on which stage did they find these particular problems.

Based on the feedback that we received from both the teams we put together all the findings we had found together in Chapter 4.

All trademarks are property of their respective owners

Here we also looked at each hypothesis we had formulated in Chapter 1, and compared them to the results we had received from the two teams in chapter 3 and prepared a final statement on these hypothesis.

As it is evident from the details in the previous chapter, each of our hypothesis proved correct and we saw that numerous benefits were achieved by the team which used a 3-layer model to build their application as compared to a team which used a 2-layer model.

Another point that was evident from this study was that the team, which used a 2-layer model, actually faced a number of problems. These are things, which could really slow down the application development and finalization process.

All trademarks are property of their respective owners

Conclusion

Now that we have completed the study we can easily conclude that we gain numerous benefits in design, development, testing and deployment when we base our application on a 3-layer UML model as compared to a 2-layer UML model. This hold true for all types of applications and especially Web based applications.

If we use a 2-layer model, not only do we loose all the benefits gained in design, development, testing and deployment which have been shown in this study but we also can run into some very potential problems as see in this study too.

All trademarks are property of their respective owners