

RT  
424  
C-1

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

**AN APPLICATION FRAMEWORK FOR THE  
DISTANCE LEARNING DOMAIN (AFDL)**

by

**AHMAD MAHER NATAFGI**

Submitted in partial fulfillment of the requirements for  
the degree of Master of Science in Computer Science

Thesis Advisor: DR. NASHAAT MANSOUR

Natural Sciences Division  
LEBANESE AMERICAN UNIVERSITY  
Beirut

January 2002

# AN APPLICATION FRAMEWORK FOR THE DISTANCE LEARNING DOMAIN (AFDL)

Ahmad Maher Natafqi

## Thesis

Submitted in partial fulfillment of the requirements for the degree of  
Master of Science in Computer Science at the

Lebanese American University

Beirut, Lebanon

January 2002



---

**Dr. Nashaat Mansour (Advisor)**

Associate Professor of Computer Science

Lebanese American University



---

**Dr. Ramzi Haraty**

Assistant Professor of Computer Science

Lebanese American University



---

**Dr. George E. Nasr**

Chairman and Associate Professor of Electrical, Computer and Industrial  
Engineering

Lebanese American University

# AN APPLICATION FRAMEWORK FOR THE DISTANCE LEARNING DOMAIN

## ABSTRACT

by

AHMAD MAHER NATAFGI

“Object-oriented application frameworks is a very important issue for the software industry as well as academia at this time when software systems are becoming increasingly complex. We believe that object-oriented application frameworks will be at the core of leading-edge software technology of the twenty-first century”. With these few words, Mohamed Fayad, Douglas Schmidt, and Ralph Johnson, a well reputable team of authors in the field of software architecture, opened their first statement on framework overview.

Software reuse has been one of the main goals for the past decades in the field of software industry, which is not a simple matter. In the past some efforts were focusing on small black-box components before the new emerging technology: Object-oriented programming. This new technology, which is based on the reuse of larger components, defined the term object-oriented application frameworks. Frameworks attracted and still many of software engineers and researches, this is due to it's advantage in increasing reusability and software productivity hence reducing time to market applications. Application Frameworks in it's own is a wide field of research for anybody to tackle. So, there was the need for selecting a domain. This selection was based on some interesting research on the domain of Distance Learning, which is quite a new subject itself. Today there are no specialist in the application of information technology to education and training. However, the domain of Distance Learning is considered to be one of the fastest growing areas on the Internet, due to the recent perception of the enormous potential for the use of Web resources for this purpose. This potentiality has attracted the attention of researchers in industry and the academic world, which are currently developing various models and products for Web-Based education and training.

To *my mother Huda Mahmoud Natafgi*  
For all the support she gave me

## ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Nashaat Mansour for his guidance throughout my M.S. studies. Thanks are also due to Dr. Nashaat Mansour, Dr. Ramzi Haraty, and Dr. George E. Nasr for being on my Thesis committee.

I would like to express my sincere gratitude to the Lebanese American University.

Special thanks to Dr. Salah Doma, for his support and care. I would also like to thank Dr. Issam Moghrabi for helping me applying to the Master's degree and for his support.

Finally, I would like to thank my family, for their long support.

## CONTENTS

<b>Chapter 1</b> .....	<b>1</b>
<b>Introduction</b> .....	<b>1</b>
1.1 Framework definition.....	1
1.2 Motivation .....	2
1.3 Problem description and definition.....	3
1.4 Overview of the proposed solution.....	3
1.5 Statement of contributions .....	5
1.6 Organization of the Thesis .....	5
<b>Chapter 2</b> .....	<b>7</b>
<b>Review of literature</b> .....	<b>7</b>
2.1 Using viewpoints to derive OO frameworks.....	7
2.1.1 Frameworks and the viewpoint-based design method .....	7
2.1.2 The Web-based education (WBE) domain case study.....	12
<b>Chapter 3</b> .....	<b>18</b>
<b>AFDL – Analysis and design</b> .....	<b>18</b>
3.1 Domain-oriented software architecture engineering framework.....	18
3.2 AFDL Introduction .....	20
3.2.1 Goals and objectives .....	20
3.2.2 Statement of scope .....	21
3.2.3 Software context .....	22
3.2.4 Major constraints .....	23
3.3 AFDL Usage scenario and use-case diagram.....	23
3.3.1 User profiles / actors .....	23
3.3.2 Use-case diagram .....	25
3.3.3 Primary and secondary scenarios.....	26
3.4 AFDL data model and description.....	27
3.4.1 Data description .....	27
3.4.1.1 Data objects.....	28
3.4.1.2 Relationships.....	30
3.5 AFDL functional model and description .....	31
3.5.1 Description of functions .....	31
3.5.1.1 Functions list and description .....	31
3.5.1.2 Performance issues.....	34
3.6 AFDL behavioral model and description.....	34
3.6.1 Activity diagram .....	35
3.6.2 Sequence diagram .....	36
3.6.3 State-chart diagram .....	37
<b>Chapter 4</b> .....	<b>39</b>
<b>AFDL – Implementation</b> .....	<b>39</b>
4.1 The five-module framework for internet application development.....	39
4.2 AFDL interface description.....	42
4.2.1 External machine interfaces.....	42
4.2.2 Human interface.....	42

4.2.2.1 The welcome page.....	43
4.2.2.2 The general information page .....	44
4.2.2.3 The virtual university page / AFDL login .....	45
4.2.2.4 The virtual university page / AFDL institution details.....	45
4.2.2.5 The virtual university page / AFDL department browsing.....	46
4.2.2.6 The virtual university page / AFDL course catalogue .....	47
4.3 AFDL restrictions, limitations, and constraints.....	52
4.4 AFDL site map (or navigation tree).....	53
4.5 AFDL hyperlinks .....	54
4.5.1 Hyperlinks for the welcome page .....	54
4.5.2 Hyperlinks for the general information page.....	54
4.5.3 Hyperlinks for the virtual university page.....	55
4.5.4 Hyperlinks for the terms of use page .....	55
4.5.5 Hyperlinks for the contact info page.....	56
4.6 AFDL architecture .....	56
4.6.1 AFDL architecture .....	56
4.7 AFDL and the observer pattern .....	61
<b>Chapter 5.....</b>	<b>63</b>
<b>AFDL – Instantiation .....</b>	<b>63</b>
5.1 Frameworks instantiation a brief definition and description.....	63
5.2 AFDL instantiation detailed description .....	63
5.3 AFDL database generation script .....	64
5.4 AFDL data initialization script.....	67
5.5 AFDL IIS configuration.....	69
5.6 AFDL miscellaneous configuration.....	70
<b>Chapter 6.....</b>	<b>71</b>
<b>AFDL – maintenance.....</b>	<b>71</b>
6.1 Frameworks maintenance: vendor viewpoint and general issues.....	71
6.2 AFDL maintenance and specific issues .....	73
<b>Chapter 7.....</b>	<b>75</b>
<b>Frequently asked questions .....</b>	<b>75</b>
7.1 Pragmatic issues behind framework design .....	75
<b>Chapter 8.....</b>	<b>78</b>
<b>Conclusion and future work.....</b>	<b>78</b>
8.1 Conclusion.....	78
8.2 List of contributions .....	79
8.3 Future work .....	80

## LIST OF FIGURES

Figure 2.1.1-1: Unification Rule .....	8
Figure 2.1.1-2: General view of the process.....	8
Figure 2.1.1-3: Viewpoint unification example.....	10
Figure 2.1.1-4: Template-hook model.....	11
Figure 2.1.1-5: Framework Design .....	11
Figure 2.1.2-6: AulaNet OMT class diagram .....	12
Figure 2.1.2-7: AulaNet site class structure.....	12
Figure 2.1.2-14: Viewpoints unification .....	15
Figure 2.1.2-20: ALADIN site class structure .....	17
Figure 3.1-1: DOSAE building phases .....	19
Figure 3.3.2-1: Use Case Diagram for AFDL .....	25
Figure 3.4.1.2-2: ERD Diagram.....	30
Figure 4.1-1: The traditional 3-layer and the 5-module framework architecture .....	40
Figure 4.6.1-2: AFDL architecture.....	59
Figure 4.6.1-3: The Five-Module Architecture for AFDL .....	60
Figure 4.7-4: Observer pattern .....	61
Figure 4.7-5: Observer pattern collaboration diagram for AFDL.....	62
Figure 6.2-1: Extension class unification .....	74



**LIST OF TABLES**

Table 3.3.3-1: Primary and Secondary Scenarios.....30

## Chapter 1

### INTRODUCTION

This work proposes an Application Framework for the Domain of Distance Learning as a strategy for overcoming current development problems encountered by the software industry in this specific domain (Distance Learning) and shows how it can be build and used through illustrating the system design diagrams in the UML model by giving a complete set of static (class and object diagrams) and dynamic (activity, interaction, state-chart, and state diagrams), views of a system design in addition to the use-case diagrams, and the deployment diagrams. In this introduction, we define object-oriented frameworks, provide the motivation behind it, we also describe some problems related to current software development habits and present application frameworks as a promising solution. We end this introduction by outlining the Thesis organizational structure, and summarizing our contributions in this work.

### 1.1 Framework definition

A framework is a reusable basic design structure expressed as a set of abstract classes and concrete classes that assists in building applications [Microsoft Press® Computer and Internet Directory, 1999] It is a reusable design for all or part of a software system; a user interface framework only provides a design for the user interface of a system, while other frameworks (like MacApp, the “Macintosh Application Framework”) provides a design for the entire application. By definition, a framework is an object-oriented design. It doesn't have to be implemented in an object-oriented language, though it usually is. Large-scale reuse of object-oriented libraries requires frameworks. The framework provides a context for the components in the library to be reused. (i.e. the structure) [Johnson1999]. Another definition of frameworks is: “a framework is the skeleton of an application that can be customized by an application developer.” (i.e. the purpose) In the second definition we define the purpose of an application frameworks while in the first definition we defined it's structure.

## 1.2 Motivation

Building a standard application differs from building an application framework. This is because while building an application framework there exists many flexible points called hot-spots that may have different implementations according to the framework instances used and are left undefined up to the actual usage time when the framework user decides to instantiate an application starting from this application framework. Where at that specific moment, the instantiation phase, the user is obliged to fill out the gaps by completing the hot-spots hence providing the specific behavior needed by the framework instances. In this sense, we think of application frameworks as a model that dictates the behavior of a group of applications belonging to the same domain. We define the kernel to represent the similarities among the concrete applications and hot-spots to represent the specific, or differences, among these concrete applications. Most of the software industry used to building applications from scratch, nowadays correct, portable, efficient and inexpensive applications cannot be build from scratch and at the same time fulfill the delivery schedule timetable. Many organizations are moving towards greater use of commercial off-the-shelf (COTS) components in the development of application systems. We believe that this strategy is not enough. A larger scale solution must be presented.

In this work we proposes a framework design that can help in the implementation of any Web-Based Educational environment application, by instantiating the proposed framework AFDL, our framework – Application Framework for the Distance Learning Domain, and filling the specific behavior for each hot-spots in it. The proposed framework design is presented using the UML model. The design of AFDL used the method of viewpoints that unifies multiple concrete applications into one global framework, which we called AFDL, Application Framework for Distance Learning. AFDL is proposed using UML model because we believe it is a universal model that is common for Software Engineering and is realized by a wide group of researchers and software developers. Moreover, we do propose a complete design for the virtual university in general and the domain of distance learning in specific. The design we propose includes a set of static views and dynamic views. In addition to that, we have implemented AFDL completely in Microsoft SQL Server version 7.0 [Microsoft SQL Server 7.0 Database Implementation – Training Kit 1999].

### **1.3 Problem description and definition**

Computing power and network bandwidth have increased dramatically over the past decade, with a relatively moderate-to-cheap cost to acquire, while on the other hand designing and implementing complex applications tends to remain expensive and prone to errors. Most of the cost and effort is due to the continuous discovery of new technology and concepts in the industry of software. Where we see that hardware architectures are growing in heterogeneity, diversity of operating system and communication platforms, all of this, contribute to difficulty and even more time in building correct, portable, efficient and inexpensive applications starting from scratch. Today, many organizations are trying hard to find a solution for this dilemma because, if they fail to do so, this will lead to failure in delivery on time, which would lead to a waste of large amounts of money and investments due to the unfulfilled contracts terms and hence losing current customers and potential markets.

Many organizations are moving towards greater use of commercial off-the-shelf (COTS) components in the development of application systems in an effort to overcome this problem. We, software engineers, believe that this strategy alone is not enough. The scale of the problem requires a larger scale solution and from here comes the role of application frameworks. [Fayad-Schmidt 1997]

### **1.4 Overview of the proposed solution**

This work focuses on providing an Application Framework for the domain of Distance Learning. In this work we show how we can help users in this domain namely researchers, and software engineers, who intend to tackle the field of distance learning, to instantiate the AFDL and use it with ease and simplicity without the need to start from scratch. For the whole idea is to use previous work and continue with it. Baring in mind that simple modification are to be made on it reflecting the hot-spots for specifically describing their needs. The solution we propose is presented using the UML universal standard model. In addition to that, an SQL notation is also presented in order for the users to directly implement the framework without any additional requirements. AFDL has been carefully studied to include a generalized version of

Distance Learning and include the concept of Virtual University. The AFDL is designed to be abstract enough to an extent that simplifies its presented diagrams and allow for several implementation approaches to be used. Moreover, tools that automate the process of instantiating the framework is also presented in this work. This is reflected in the SQL database structure generation scripts and database initialization scripts, framework itself with its embedded components such as the FTP utility, Query builder, and Search components.

Tools presented in this Thesis are developed using Microsoft SQL Server version 7.0 and can be easily upgraded to Microsoft SQL Server 2000. In addition to that, Borland's Delphi development environment version 5.0 has been also selected and can also be easily upgraded without any additional modifications to the latest version 6.0. (Later we will explain the basis for our selection). These tools include:

- *AFDL Framework*: This includes the framework itself represented as an ActiveX control.
- *Database Generation Script*: This includes SQL scripts or basically stored procedures that once executed would generate a complete and comprehensive database for the domain of Virtual University in general and Distance Learning in specific.
- *AFDL Site*: This includes the framework online site which will be presented to the final end user as a service provided by the hosting server.
- *FTP Tool*: File Transfer Protocol tool that handles all file transfers from and to the server providing services to the user on the Web.
- *Query Builder Tool*: Query builder, helps the user to build any form of query to be used later as an input for execution on the database in order to return result on any query he might need to know with simplicity.
- *Database Initialization Scripts*: This includes a need to have data in the database generated by the previously defined script. An example of such need to have data include schemas such as the country table, the city table, state table, and other similar fixed data input.

Of course, in order for the application developers to provide additional assistance in maintaining and verifying the framework can use the development environments discussed previously.

## 1.5 Statement of contributions

The major contributions of this work are:

- *AFDL Framework*: This includes the framework itself represented as an ActiveX control, which includes an implementation of the functional specification for the Distance Learning domain.
- *UML Diagrams*: A UML representation of the Distance Learning domain. This includes a complete set of static and dynamic diagrams defined and ready-for-use to the application developers in this field.
- *Distance Learning Database Structure*: This includes a complete structure for the Distance Learning database.
- *General Domain-Specific Tools*: These include: File Transfer Protocol tool that handles all file transfers from and to the server providing services to the end-users on the Web, Query builder that helps the end-user to build any form of query to be used later as an input for execution on the database in order to return result on any query he might need to know with simplicity.
- *Database generation script*: An automated database generation script defined in Microsoft SQL Server. In which the database can be automatically generated in any Distance Learning computer server machine. Hence, simplifying the process of AFDL instantiation and implementation.

## 1.6 Organization of the Thesis

The next chapter, chapter two, gives background information. It presents common research in the area of Application Frameworks in general and in Distance Learning in specific. Such these research areas are: viewpoints method to derive object-oriented frameworks, Framework design and instantiation based on viewpoint method, ALADIN.

Chapter three presents a complete analysis and design model for Application framework in the domain of Distance Learning. It includes the general functional

specifications regarding both sites the authoring site and the learning site. It also provides a complete set of diagrams (Static and Dynamic) in the UML model. These diagrams basically includes a use-case diagrams, class diagrams, interactive diagrams, activity diagrams, and the deployment diagram.

Chapter four presents the actual AFDL implementation interface description, site map (navigation tree), hyperlinks, and all related work.

Chapter five presents a detailed guideline for the AFDL instantiation and may be used as a documentary reference for the framework end-user.

Chapter six presents a detailed maintenance guideline for the AFDL.

Chapter seven presents some of the frequently asked questions for the AFDL.

Finally, chapter eight provides a conclusion to the Thesis, list of contributions provided by us and highlights issues not tackled in this Thesis and future work for future researches to focus on.

In Appendix-A we provide an SQL notation version of the complete database design as a practical ready-for-use implementation.

## Chapter 2

### REVIEW OF LITERATURE

In this chapter we provide an overview of previous literature in relation to application frameworks in distance learning. In this regard, we state a review of previously used methods to derive Object-Oriented frameworks in the domain of web-based education. In addition to that, we review the five-module framework for Internet application development, and Domain-Oriented Software architecture-engineering framework.

### 2.1 Using viewpoints to derive OO frameworks

In his paper “Using viewpoints to derive an object-oriented frameworks”, Fontoura introduces a new method for structuring object-oriented frameworks based on the analysis of a set of existent applications. These applications are defined as viewpoints of a domain, and rules are applied to derive the domain abstractions from viewpoint definitions. The method applicability is illustrated by a large real case study in the Web-based education (WBE) domain.

Section 2.1.1 below, presents the software engineering concepts that serve as a basis for the viewpoints method. Section 2.1.2, describes the viewpoint-based design method and a supporting environment used to systematize the method application.

#### 2.1.1 Frameworks and the viewpoint-based design method

The following software engineering concepts are being presented:

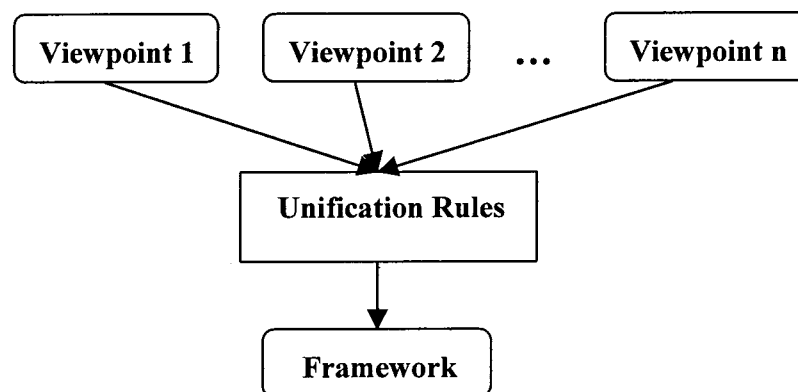
**I- Frameworks:** The viewpoint-based method addresses the problem of previously used object-oriented design methods that neglect the importance of helping the framework designer to structure the system’s kernel and hot-spots.

**II- Viewpoints:** The development of complex software systems involves many agents with different perspectives (or **viewpoints**) of the system they are trying to describe.

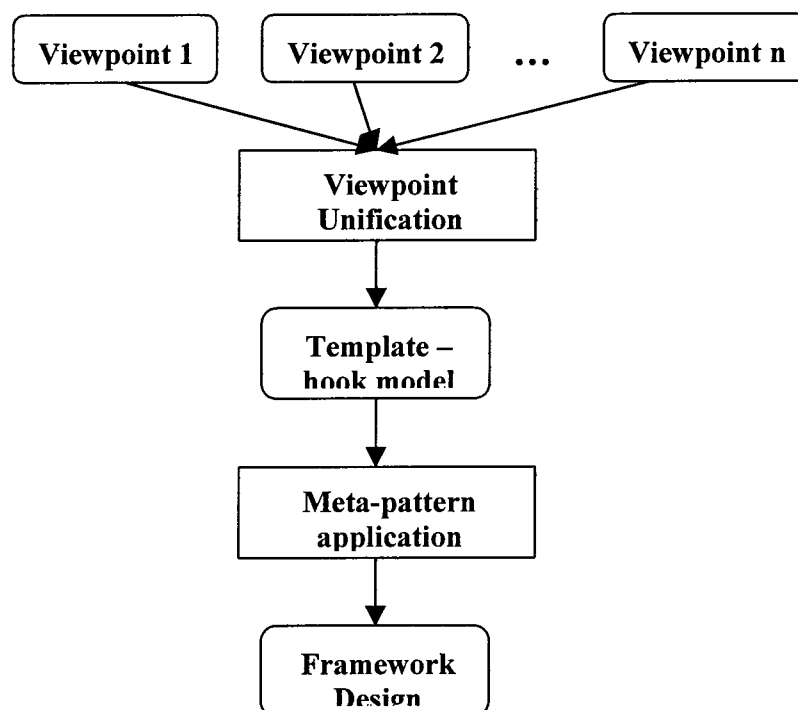
**III- General description of the viewpoint-based design method:** “Developing reusable frameworks cannot occur by simply sitting down and thinking about the



problem domain. No one has the insight to come up with the proper abstractions". The viewpoint-based design method proposes the development of concrete examples in order to understand the domain. The viewpoint-based design method allows the definition of frameworks through the analysis of concrete applications, which are defined as viewpoints of the framework domain. In this regard, the domain is a set of viewpoints, and every domain has an associated framework. Unification rules are defined as a way of describing how applications, or viewpoints, can be combined to compose a framework. (See **Figure 2.1.1-1** below). [Fontoura-Crespo-Lucena-Alencar-Cowan 1999]



**Figure 2.1.1-1:** Unification Rule



**Figure 2.1.1-2:** General view of the process

**IV- Viewpoint unification:** Some preconditions, on the viewpoints that are going to be unified, are presented:

1. Same concepts must be represented by the same names
2. Orthogonal concepts
3. Attribute types must be consistent
4. Cyclic hierarchies must be avoided

When the set of viewpoints is consistent unification rules can be applied. The result of the unification process is a template-hook model. The unification process is based on the following rules:

1. Every class belonging to the set of viewpoints has a corresponding class, with the same name, in the template-hook model.
2. If a method has the same name, signature, and implementation in all the viewpoints it appears, it has a corresponding method, with the same name, signature, and implementation, in the template-hook model.
3. If a method exists in more than one viewpoint with different signature, it has a corresponding hook method in the template-hook model, with the same name but undefined signature and implementation.
4. If a method exists in more than one viewpoint with different implementation, it has a corresponding hook method in the template-hook model, with same name and signature but undefined implementation.
5. All the methods that use hook methods are defined as template methods. There is always a hot-spot relationship between the class that has a template method and the class that has its correspondent hook method.
6. All the existing relationships in the set of viewpoints that have no corresponding hot-spot relationship are maintained in the template-hook model.

**Figure 2.1.1-3** shows an example of two consistent viewpoints that are to be unified. In Viewpoint 1, the main class is *Report*, which uses classes *GetData* to access the information required by the report and *GenerateReport* to generate the final report in an HTML file. Class *GenerateReport* uses class *FormatReport* to configure the layout of the generated HTML file. Notice also that in Viewpoint2, we also have the same method *GenerateReport* that generates the final report in a RTF format (i.e. it

does have different signature). In this case rule 3 implies that *GenerateReport* is a hook method. Rule 4 defines method *FormatReport*, which has different implementations in the 2 viewpoints → a hook method. Rule 5 defines all methods that use hook methods to be template methods. Thus the template methods in this example are *Report* and *GenerateReport*.

**Figure 2.1.1-4** shows the result of the unification of viewpoints. Note that the signature of method *GenerateReport* is UNDEFINED as established by rule 3, and the implementation of the methods *GenerateReport* and *Format* are also UNDEFINED as established by rule 3 and 4. All undefined signatures and implementations are to be defined by the framework user upon instantiation.

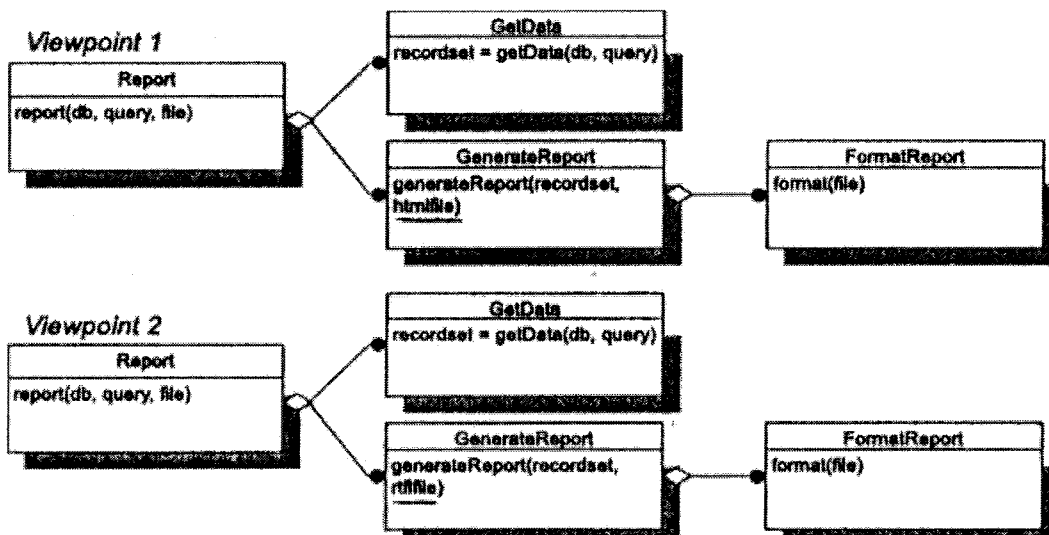


Figure 2.1.1-3: Viewpoint unification example

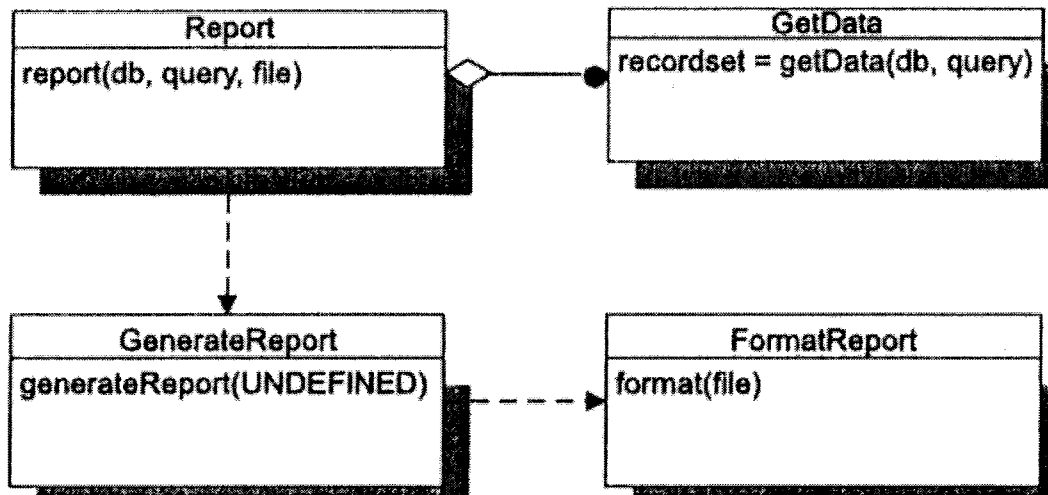


Figure 2.1.1-4: Template-hook model

[Fontoura-Crespo-Lucena-Alencar-Cowan1999]

**V- Applying meta-patterns:** The framework designer must eliminate all hot-spot relationships from the template-hook model, replacing them by the most appropriate meta-pattern. The result diagram after meta-patterns application is the framework design in **Figure 2.1.1-5**.

Since the meta-pattern used for the hot-spot relationship between classes *GenerateReport* and *FormatReport* unifies the template and hook methods in the same class, class *FormatReport* is not required in the framework design. A very important property of the method is the control of the design complexity, leading to simple and readable designs. In this example, the generated framework design has fewer classes than each of the original viewpoints.

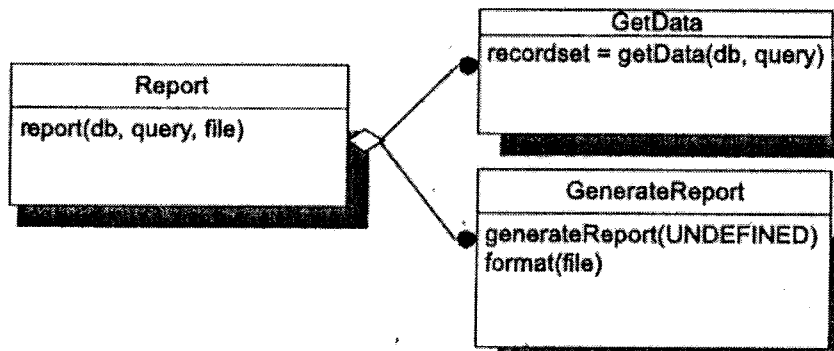


Figure 2.1.1-5: Framework Design

### 2.1.2 The Web-based education (WBE) domain case study

This section presents the models of the analyzed Web-Based Education (WBE) environments and describes the application of the viewpoints-based design method to generate the **ALADIN** framework, an application framework presented by Fontoura. Six WBE environments are used for the case study. Their selection is based on their (i) popularity and importance, (ii) availability of documentation, (iii) and their underlying concepts, so that, environments that do not model new concepts are discarded. The following six viewpoints will be illustrated with the use of figures.

1. AulaNet
2. LiveBooks
3. Web course in a box
4. Web-CT
5. LearningSpace and Virtual-U

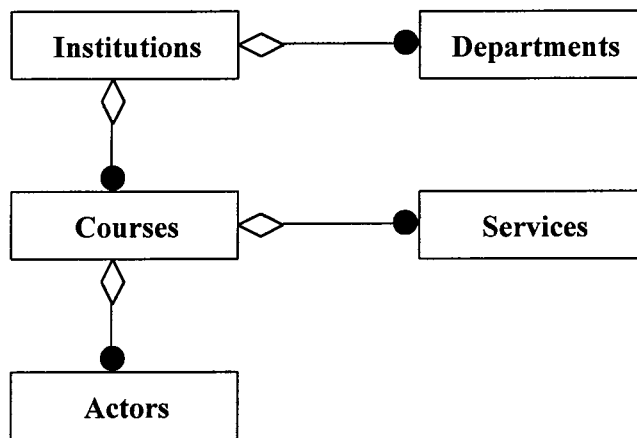


Figure 2.1.2-6: AulaNet OMT class diagram

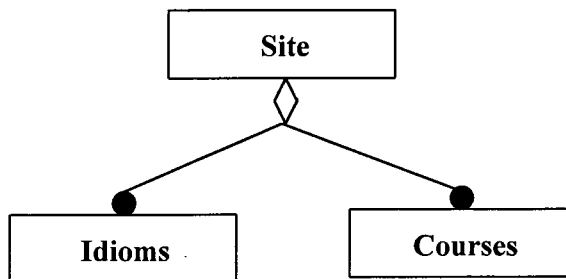


Figure 2.1.2-7: AulaNet site class structure

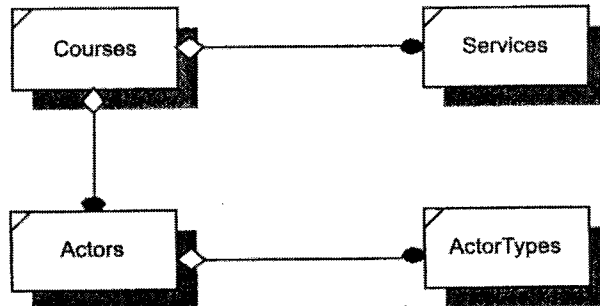


Figure 2.1.2-8: LiveBooks class diagram

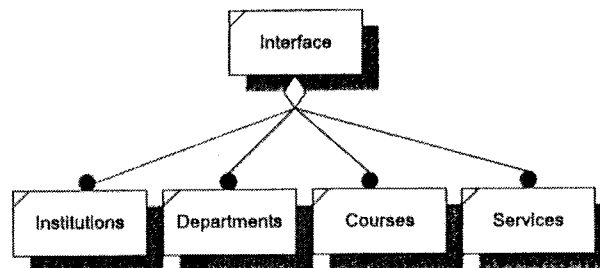


Figure 2.1.2-9: WCB interface class structure

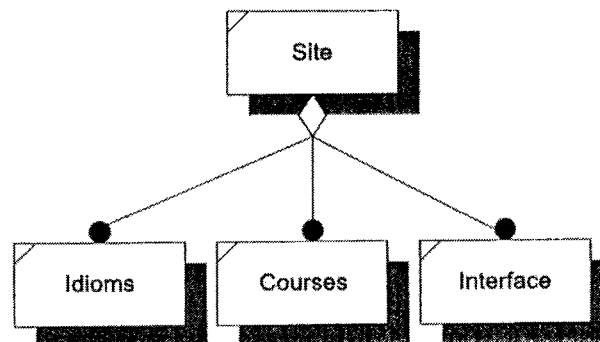


Figure 2.1.2-10: WCB site class structure

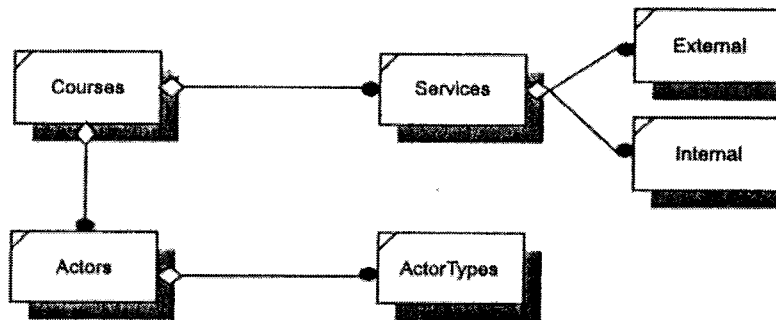


Figure 2.1.2-11: Web-CT class diagram

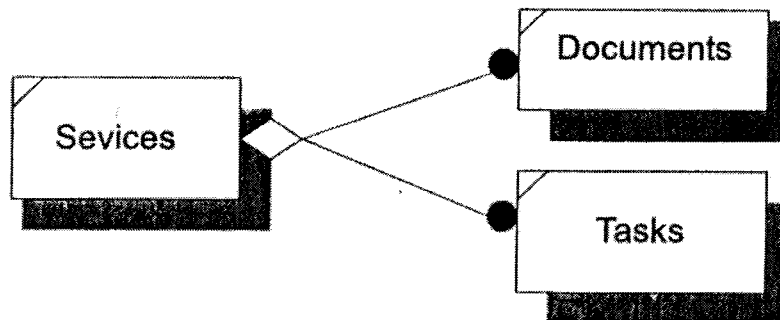


Figure 2.1.2-12: Learningspace and Virtual-U services class structure

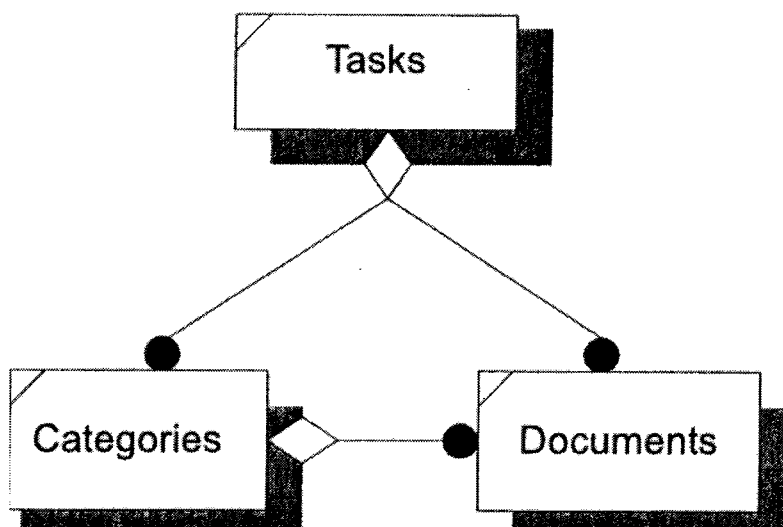


Figure 2.1.2-13: LearningSpace: tasks, categories, and documents

The viewpoint unification figure is as follows:

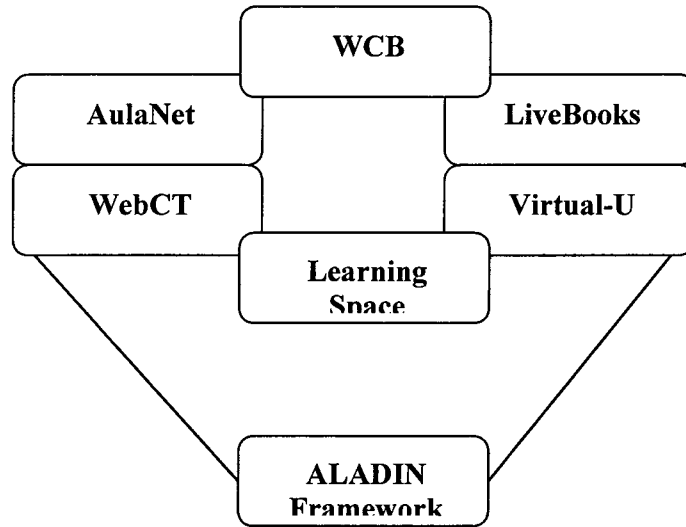


Figure 2.1.2-14: Viewpoints unification

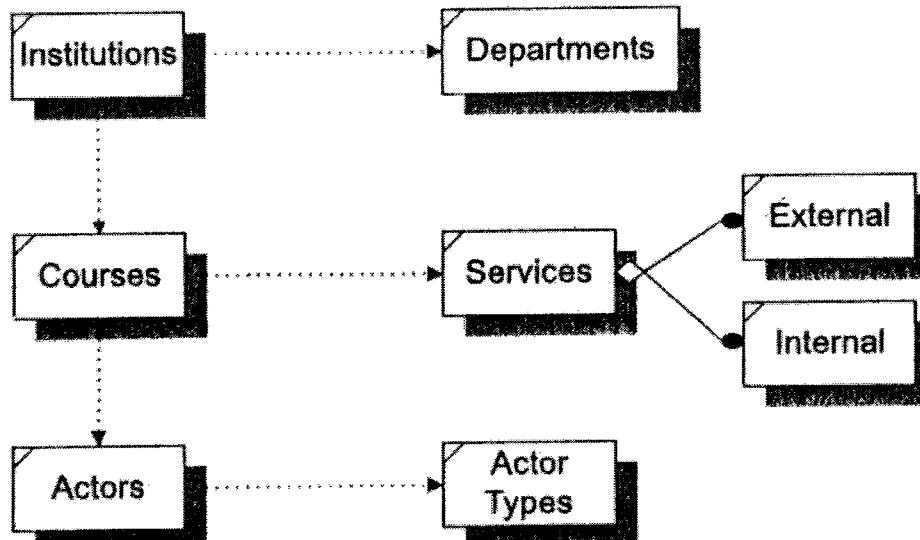


Figure 2.1.2-15: ALADIN template-hook model



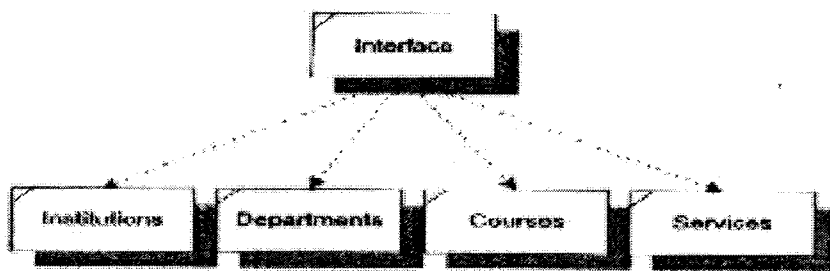


Figure 2.1.2-16: ALADIN interface template -hook model

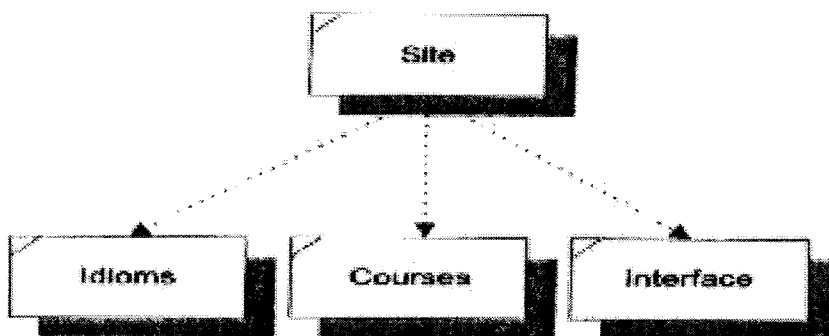


Figure 2.1.2-17: ALADIN site template -hook model

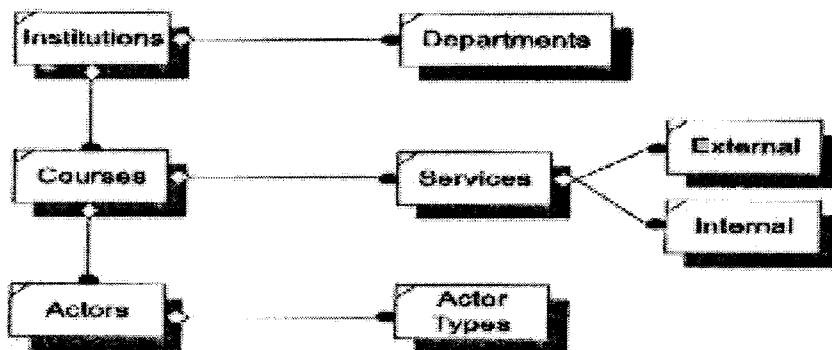


Figure 2.1.2-18: ALADIN class diagram

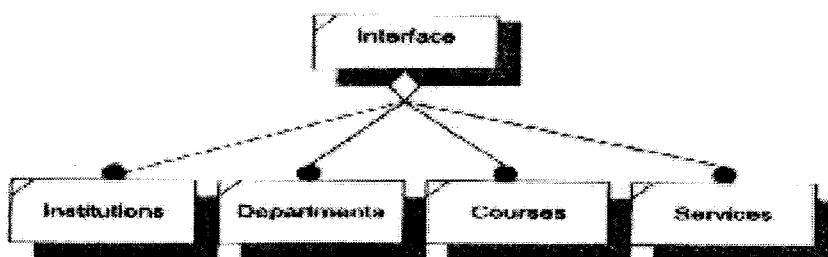
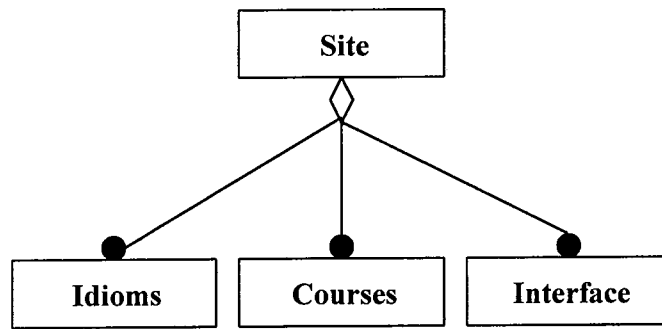


Figure 2.1.2-19: ALADIN interface diagram



**Figure 2.1.2-20:** ALADIN site class structure

To conclude, the basic idea was to identify a framework that could provide all the functionality required for generating, at least, the six analyzed environments. And as you can see the final generated design is the ALADIN framework that includes all of the six viewpoints. [Fontoura-Heausler-Lucena1998]

## Chapter 3

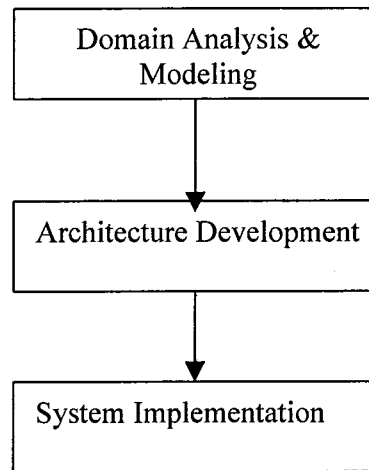
### AFDL - ANALYSIS AND DESIGN

In this chapter we provide a detailed and complete analysis and design model for the Application Framework we are proposing for the domain of distance learning. But before doing so, we provide an overview of the reasons behind selecting a specific domain for building an application framework, and what are the steps to be followed in doing so i.e. from where to start and what phases are to be respected.

#### **3.1 Domain-oriented software architecture engineering framework**

In this section, we refer to a paper that provides a detailed description of Domain-Oriented Software Architecture Engineering Framework (DOSAE), an environment being developed by Software Productivity Solutions, Inc. that provides a domain framework that supports the development of systems based on an architectural view of the systems. It highlights the steps - and rationale for each step – required to build a domain architecture framework. Moreover, the author gives three main reasons for following a domain-oriented approach when building a framework. He states that a domain-oriented approach is being followed because it:

- Will integrate with domain modeling efforts that are precursors to system development activities
- Acknowledges the similarities in construction of applications within a given domain resulting in an improved opportunity for reuse over other reuse approaches
- Leverages the expertise that developers accumulate when working on a product line of systems that fall within a given domain.



**Figure 3.1-1:** DOSAE building phases

In **Figure 3.1-1** above, we see three separate highly related engineering processes: Domain Analysis & Modeling, Architecture Development, and System Implementation. Each of these processes is composed of one or more activities. The Domain Analysis & Modeling process defines an organization's overall understanding of a problem domain area. Domain Analysis identifies the information, functions, relations, constraints, etc., that are relevant to the selected domain. In this regard, in section 3.2, we provide a detailed domain analysis and design for **AFDL**. By creating an object-oriented representation of the domain, Domain Modeling serves as a focal point for agreement among various participants, stimulates further analysis, documents the domain, and is an input to the Architecture Development process. Architecture Development aims to create architectures that guide current and future system development efforts. The Architecture defines the overall structure and organization of a software system. Two kinds of architectures are created: The first is a domain architecture that represents a template for building multiple, similar systems within the given domain. The second is a system-specific architecture that is to be used as the initial design of a system to be implemented. Development of either a domain or system-specific architecture identifies architectural abstractions in the form of:

- Architectural styles and patterns used to define organizational structures within the architecture
- Architectural components identifying the functions that are to be provided within the system
- Architectural constraints within place limitations on allowed component interactions or define non-functional properties

We believe that working from an architectural framework greatly enhances the opportunities for reuse [Krause1997]. In this regard, in Chapter four, we propose an architecture structure for the **AFDL** and we discuss in details the implementation phase.

## 3.2 AFDL Introduction

### 3.2.1 Goals and objectives

This work proposes an Application Framework for the Domain of Distance Learning (**AFDL**) as a strategy for overcoming current development problems encountered by the software industry in the specific domain of Distance Learning. We aim at building an application framework that can be instantiated in the distance learning domain with little effort as possible to be made in modifying the specific details (or hot-spots) related to the business. **AFDL** framework is expected to include a generalized view of the distance learning field reflected in it's persistent classes in the main database, also reflected in it's main components to be available to the end-user, functionality, and services. These are called the common or kernel specifications. We expect that **AFDL** would increase highly the profits of the business implementing it by decreasing the time process for development and increasing software reuse and concentrating on the market issues rather than the development and technical aspect of the business.

**AFDL** should be used by any industry whose business is in the distance-learning field. Universities, colleges, high schools, training centers, even publishers, and book authors to name a few. **AFDL** is a web-based application framework that provides services online using the Internet connectivity through out the world to any user

(student) anywhere in this world and at any time provided he/she is connected to the web.

**Note:** that we are not interested in the flexible points called hot-spots that may have different implementations according to the framework instances and hence the hot-spots are left undefined up-to the actual time when the framework is instantiated.

### 3.2.2 Statement of scope

**AFDL's** - Application Framework for the Distance Learning domain, is a web-based framework that can be instantiated by developers in the same field of business of eLearning. Once Instantiated, the developers can add the implementation parts specific to their business domain and interests. These parts are called the hot-spots. After completion, the final application must establish a set of learning objectives available for the end-user. **AFDL's** - Application Framework for the Distance Learning domain, final end users can be categorized under five major groups/or actors: The *Instructor*, the *student*, the *author*, the *teaching assistant*, and the *administrator*. Each of these actors has different roles assigned to them of which some of these roles may be in common. (To be discussed in details in the Use-Case Diagram).

The major inputs to the final instantiated system include mainly: information about the institution with it's departments, faculty members, students profiles, majors offered, courses offered, and for each course it's related curriculum offered, which includes the selected textbooks, topics, and materials to be covered. This information can be entered into the system in several formats. For example, it can be entered in a scanned document format (.Doc), as a text format (.txt), as an audio/visual format (.avi), as a sound format (.wav or .mp3), as a browser-compatible format file (.html) ...etc.

The basic functionality to be offered includes a set of tools available to the intended users. For example, the author is provided with a file transfer utility (FTP tools) that helps him in uploading, and downloading materials in to the database. These sets of functionality basically includes, Database generation scripts, Database initialization scripts, the FTP Tool, Query Builder, Auto-Linker, the framework itself with all of

it's built in general routines and functions, including searching into the material, and the AFDL site.

The basic output results are reflected in the ability to browse all of the information available in the database on the screen in a user – friendly interface (using any Internet browser) in multiple views according to the user group. (This includes an author view, instructor view, student view, teaching assistant view, and administrator view). For each view, different set of services is provided and enabled. (To be discussed later in details.)

### **3.2.3 Software context**

As discussed briefly above, AFDL framework in general is being build to serve the eLearning (Electronic Learning) industry as a whole. AFDL is a web-based application framework. It is a semi-complete\* solution provided online using the connectivity of the web (Internet). Hence, any user (student) connected to the Internet can use AFDL services. So, no geographical limits, moreover, it is available online 24hours/day over 7 days/week, 365 days/ year. The user can connect any time, anywhere on his free will, no fixed class schedules, no restrictions, no office hours, and no obligations what so ever. In addition to that, all of the new technology services are provided to the end-user in addition to the traditional class services. For example, power-point presentations, online references browsing, discussion groups, search utility, downloading a soft copy of the materials (for hard print outs), online illustrations with diagrams, pictures, and videos, in addition to the video conferencing (collaboration) all are available just one-click away.

The eLearning industry may include a whole set of universities, colleges, high-schools, training centers, even publishers, and books authors who are interested in providing there books, materials, curriculums to a wide range of users include those who are geographically distant. Hence the potential customers are wider in range. For example, no attendance is required, no exams are required (assessments are up-to the user for in case of certification request or for self-assessment purposes), references are just a one-click away ...etc.

\*: By semi-complete we mean that there is still the final instantiation stage to be completed by filling out the specific implementation details related to the business itself (that we refer to by hot-spots).

### 3.2.4 Major constraints

AFDL's main implementation constraint is related to the constraints inflicted by the World Wide Web (or Internet) itself regarding the speed of communication and file transfer speed (either uploading or downloading). In case the framework is to be used via a high-speed connection links such as an ISDN or higher connection speeds (128K and above), this constraint would be not effective any more.

In addition to that, one additional constraint may be imposed by the fact that our AFDL is a framework and not the final software. Which bares both benefits and constraints at the same time. Regarding the constraints, well, a framework is an unfinished work. I.e. the Hot-spots are still waiting to be implemented usually by experienced developers. Also, these experienced developers need to pass through our documentation and framework components and get to know them before they are able to fully benefit from the instantiation process.

## 3.3 AFDL Usage scenario and use-case diagram

This section provides a usage scenario (or the software-side) for the framework in addition to a complete use-case diagram. We organize the information collected during requirements elicitation into use-cases.

### 3.3.1 User profiles / actors

In our implementation of the AFDL, we envision five actors: the *author*, the *instructor*, the *teaching assistant*, the *student* and the *administrator*.

1. **Author:** is responsible for creating instructional material for the course. This material will be the core of an electronic textbook, organized along the principles outlined below.



2. **Instructor:** is responsible for organizing the instructional material for a particular course given to a specific set of students. Presumably the instructor organizes the material based on his or her instructional style and the needs and capabilities of the particular class. The instructor is free to use and adapt the material provided by the author and can supplement this material with that of his/her own. The instructor is responsible for setting the objectives for a particular course.
3. **Teaching assistant:** is responsible for analyzing the progress of the students and providing feedback to the instructor about each student's performance.
4. **Student:** is responsible for the real job of learning.
5. **Administrator:** facilitates integration between instructor, course, and the student. He deals with matters like enrolling students, making course news, ...etc.



### 3.3.3 Primary and secondary scenarios

Special requirements associated with the use of the software are presented below.

**Table 3.3.3-1:** Primary and secondary scenarios

<i>No.</i>	<i>Actor Actions</i>	<i>System Response</i>
1.	The Administrator opens accounts / manage profiles for the students, instructors, and authors and teaching assistants.	Users are created
2.	The Administrator publishes/creates course catalogue.	Course catalogue is created
3.	Instructor selects course to teach.	
4.	Instructor prepares topics to be covered in course	
5.	Author creates topics into the database	Topics database is created for the specific course
6.	Instructor prepares materials for each course tutored	
7.	Author creates materials into the database	Material database is created for the specific course
8.	The Student Log in to the system remotely	
9.	The Student selects the interface language	User interface changes acc. To selected interface language
10.	The Student registers and pays tuition	
11.	The Student selects the department/course to learn	
12.	The Student browses topics and materials	
13.	The Student undertakes assessment quizzes	Assessment response is marked into the system
14.	The Teaching assistant grades students assessment	Teacher Assistant updates grades
15.	The Instructor evaluates/monitors progress report	
16.	The Student browse his grades report	System displays the grades report
17.	The Student selects different services	Establish service ex. Video conference connection
18.	The Student creates/edit his profile information	
19.	The Instructor manages respond to student emails	

#### *Alternatives*

- Step 8: Student fails to log in due to unsettled payments or unregistered user.
- Step 17: A Service might fails to establish connection at a specified minimum speed due to connectivity problems or connection pressure.

## 3.4 AFDL data model and description

The major subsystems and databases used in the current implementation of **AFDL** are the *distance learning* database which includes a set of schemas – tables that are inter-related to each other according to the well known concept of relational databases. The major schemas or tables that are defined in the *distance-learning* database are presented in this section. Moreover, they are implemented using the well-known and used Microsoft SQL Server version 7.0, which could be easily upgraded to version 2000. This section describes the information domain for the **AFDL**.

### 3.4.1 Data description

The major data objects that will be managed/manipulated by the **AFDL** are described in this section. Note that the emphasis here is on the major data objects (persistent tables) only. They represent the core of the database system.

**Course:** consists of course information and includes the following fields: number, code, name, description, credits, prerequisites, syllabus.

**Topic:** The subject material is organized as a set of information topic explanations, learning tasks, examples, and tests implemented as HTML pagelets. Each item in the database is indexed in the following ways:

- Major Topic/Subtopic: this is the primary search key.
- Level: easy, average, advanced
- Type: Explanation, Learning task, Example, Test

In addition, each entry has the URL of the actual content material. It consists of the following field: CourseID, ParentID, Interface, Title, Description.

**Material:** The curriculum is defined as HTML web pages with hyperlinks to associated information from the topics database. Each HTML page can correspond to a chapter in a traditional textbook, to a lecture that might be drawn from several information sources or to exercises and projects for the student. Such curriculum material can be generated by the author based on an understanding of the subject material or by the instructor based on personal instructing style, an understanding of student population, goals of the academic program, and

institutional policies. It includes the following fields: TopicID, Type, Interface, Content, Text.

**Service:** it consists of the services provided by the system to the students about the materials presented. Basically these services include, displaying information in different formats, HTML, Powerpoint, AVI, Wav or mp3, etc...

**Student:** it consists of academic information about the student (class, section, major, minor); his personal information can be obtained from the common person schema-table.

**Faculty:** it includes additional academic information about the faculty member himself (phone, email, cv, officehrs, rank, salary, chairs, advisor, dean).

**Person:** is a generic table for all personnel and students inside the institution. It includes all personal information details: First, Last, IDType, IDNo, Faculty, Student, DateOfBirth, Gender, Address and etc...

#### 3.4.1.1 Data objects

A detailed and complete list of the data objects or tables (schemas) and their major attributes are described in bellow:

1. **Student:** It holds the academic details of the student, Student attributes are: ID, Class, Section, Major, and Minor.
2. **Class:** It includes the class levels available in the institution. Its attributes are: ID, Description.
3. **Major:** It includes the list of major available by the selected department in the selected institution. Its attributes are: ID, Description, Department, Degree, and DegreeSymbol.
4. **Minor:** It includes the list of minor available by the selected department in the selected institution. It's attributes are: (ID, Department, Description)
5. **Institution:** It is used to select from a list of different institution in case the business industry supports multiple institutions. It's attributes includes: ID, COffice, Cname, Dean, Address, POBox, Phones, Fax, Email, Site, Note
6. **Department:** It is used to select from a list of available departments under the selected institution. Its attributes are: ID, Dname, Institution, Dphone, and Office.
7. **FacultyMember:** (For future enhancement)
8. **Course:** It includes details about the course including: ID, Cnumber, Interface, Code, Cname, Department, Cdesc, Credits, Prerequisites, and Syllabus.
9. **Topic:** It includes the topics to be covered in a selected course in order to be academically fulfilled. Its attributes are: ID, CourseID, ParentID, Interface, Title, and Description.
10. **Material:** It includes the detailed material to be covered indexed according the topics. Its attributes are: ID, TopicsID, Type, Interface, Content, and Text.

- 11. MaterialType:** It includes the different types of materials available to the end-user in different electronic formats. Its attributes are: ID, Code, Description, and Extension.
- 12. Interface:** It is used to support multiple languages and look and feel as an interface for the user. It's attributes are: ID, Language, SiteType
- 13. SiteType:** It is used to differentiate between a learning site type and the authoring site.
- 14. Site:** As SiteType table, it is used also to differentiate between sites with there URL addresses. It's attributes are: ID, Type, Address.
- 15. Services:** It is used to list the different supported services available to the end-user. Its attributes are: ID, Description, Type, Interface, and Course.
- 16. ServiceType:** It's attributes are: ID, Description
- 17. Section:** It is used to hold information about the student's sections registered currently and in the past. It's attributes are: ID, SecNumber, Course, Year, Semester, CurrentSection, Student, Instructor, EnrolledStudents, AvailableClass
- 18. InstructorResearcher:** (For future enhancement)
- 19. Support:** (For future enhancement)
- 20. Grant:** (For future enhancement)
- 21. GradStudent:** (For future enhancement)
- 22. Degree:** (For future enhancement)
- 23. Advisor:** It is used to select the faculty members whom are at the same time advisors. Its attributes are:
- 24. Semester:** It is used to list available academic timetable for the institution.
- 25. Transcript:** (For future enhancement)
- 26. Committee:** (For future enhancement)
- 27. IDType:** It is used to provide a list of different types of identification recording in the system for a specific user. Its attributes are: ID, Description.
- 28. Person:** its attributes are: Id, FirstName, MiddleName, LastName, MotherName, IDType, IDNumber, Faculty, Student, DateOfBirth, Gender, Address etc...
- 29. Area:** It holds information available about areas in a selected country.
- 30. State:** It holds information available about states in a selected country.
- 31. Country:** It holds a list of all countries in the world.
- 32. City:** It holds a list of the major cities for each country.
- 33. Faculty:** It holds the major faculty members.
- 34. Rank:** It holds the rank of the faculty members.
- 35. User:** It holds a list of registered users (could be author, instructor, administrator, students, learning assistants) and their corresponding encrypted passwords. It is basically used for security reasons. Its attributes include: ID, UserName, Password, and UserGroup.
- 36. UserGroup:** It is used for security reasons. Its attributes are: ID, Description.
- 37. AccessLevel:** It is used for also security reasons. Its attributes include: ID, UserGroup, Table, TableRead, TableWrite, StoredProcedures. It defines the access level available for a selected group of users.

### 3.4.1.2 Relationships

Relationships among data objects are described in this section using an ERD Diagram. It is presented in the final format as it appears in the MS-SQL Server diagram. It includes all of the schemas-tables used in the system.

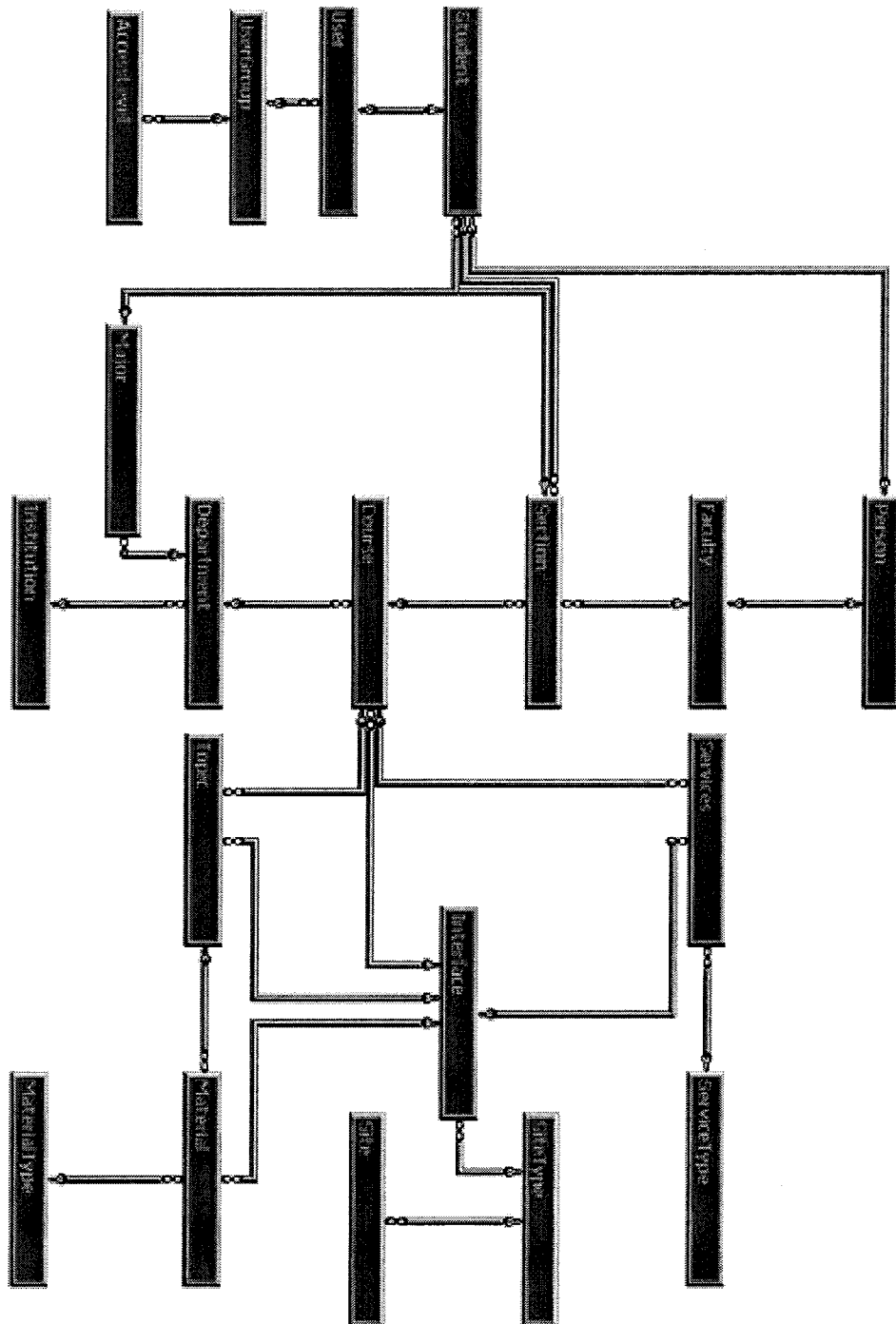


Figure 3.4.1.2-2: ERD Diagram

## 3.5 AFDL functional model and description

A description of each major software function, along with an interactive diagram is presented.

### 3.5.1 Description of functions

In this section we present a detailed description of all of the functionalities of the AFDL.

#### 3.5.1.1 Functions list and description

1. **Database generation scripts: AFDL**, as intended to be is a framework that includes a predefined and implemented set of services and functionalities including of course database generation tools. One of the major tools supported by MS-SQL Server version 7.0 to enable generation of the database is through the use of the script generation tool (utility). Regarding this functionality, we are presenting to the business industry the framework with an easy to instantiate tools – The Database generation scripts. It consists basically of a Transact - SQL script that includes in it commands used by the MS-SQL Server in order to generate a full and complete design of the Distance Learning database. This includes of course a list of all tables with their attributes, data type for each attribute, size, default value, min, and max values, ...etc. Moreover, it includes also the relationship between each and every schema / table. In short, it generates the complete set of data objects related to the Distance Learning domain. (No additional modifications are required in this functionality from the application developers who are instantiating AFDL.
2. **Database initialization scripts:** Are a Transact – SQL scripts that are provided to the business industry built-in with the framework itself in order to generate an initial set of data that are required. For example, when we talk about the schema named country, we already presuppose that it includes a list of all the countries in the world. This is basically what functionality Database initialization scripts provide. It initiates data with the basic and need to have data. Data has been collected from different sources on the Internet and reviewed for syntactic errors. It is not intended to be a complete list. But just a



starting point. In addition to that, a demo data has been embedded in these database generation scripts for demonstration purposes only.

3. **FTP Tool:** File transfer Protocol tools, it provides a tool or utility program to upload and download files to and from the web site that provides these services to the user. It includes connection services by providing the IP address of the server site, with its port, and a lot of other embedded detailed features.
4. **Query Builder:** This tool is a very powerful component and one of the most important one that must be provided with every framework in general that communicates with a database. It provides the end-user with a user-friendly tool/utility that helps him in building any query required by him for the computer to search into the database and retrieve the results. It should be designed in a way to support building the syntax of a Transact-SQL graphically.
5. **Auto-Linker:** provides a quick access to any referenced topic on any HTML page, an auto linker which takes every topic in the subject database and replaces every reference to that topic by a hyperlink to the topic. The automatic linking of topics is done periodically on every HTML page in the website and upon addition or modification of the content of any topic in the database (it should update the related subjects to the topic automatically - this could be done by building a tree structure, with the topic being the parent node and it's children it's related topics and all of it's sub-related topics). Note here, that we are interested in the direct parent node children only. The primary advantage of auto-linking is that the web author or instructor is not burdened with manually linking topics with their references. Also topic explanations added after a page referencing that topic will be linked to the newly added page without the original author being aware of the new addition.
6. **Glossary:** The **AFDL** in intended to provide a search engine over the entire website and a glossary of terms.
7. **Multi Interface:** provides the end-user, students, authors, instructors, teaching assistant, and administrators with a multiple interface selection. (Multiple views).

8. **Multi Lingual:** provides the end-user, students, authors, instructors, teaching assistant, and administrators with a multiple languages selection by providing the capability to select from different available languages.
9. **Online Assessment:** Automatic online assessment is to be provided for the end-user (student) to track student progress. With a graphical display showing the areas of need-to-improve for future studies.
10. **Online references:** links to libraries, international conferences, published papers and publishers.
11. **Faculty Catalogue:** it displays different faculty member names and position in the selected institution and their contact information.
12. **Course Catalogue:** it displays all courses provided by the selected institution with their related department. It provides the students with the course number, name, description, number of credits, and prerequisites. It serves the students in their registration period.
13. **User Group Level Security:** provides an advanced level of security to the system by providing all users belonging to the same group an equal privilege and hence an equal set of services available to each user belonging to this group. In addition to that multiple user groups are defined and created with each having a set of privileges and access rights created, maintained and defined by the administrator.
14. **Access Security:** It defines the access rights assigned to user groups including browsing, writing, uploading, downloading, and any format of accessing any information. It is related directly to the user group table.
15. **Multiple Services:** This includes multiple presentation formats for the materials in the database. It includes power-points presentations, online collaboration (video conferencing), audio conferencing, and rich text format display (HTML pagelets).
16. **Filter Component:** Provided to enable the end-user to filter selections out from the database retrieval.
17. **Collaboration interaction:** includes video conferencing and chatting.
18. **Tree View Topic Display:** includes a tree view like display window that lists all the topics to be covered for a specified course.

19. **Multiple institutions:** it provides the business industry with the ability to service multiple teaching institutions at the same time. Where for each institution, a complete database system is implemented.
20. **Complete system design:** By a complete system design we mean that the Distance Learning database design includes a complete data objects designed to provide additional functionalities for a universal University.
21. **Reporting:** It should provide a list of reports available for students, authors, instructors, and teaching assistants.

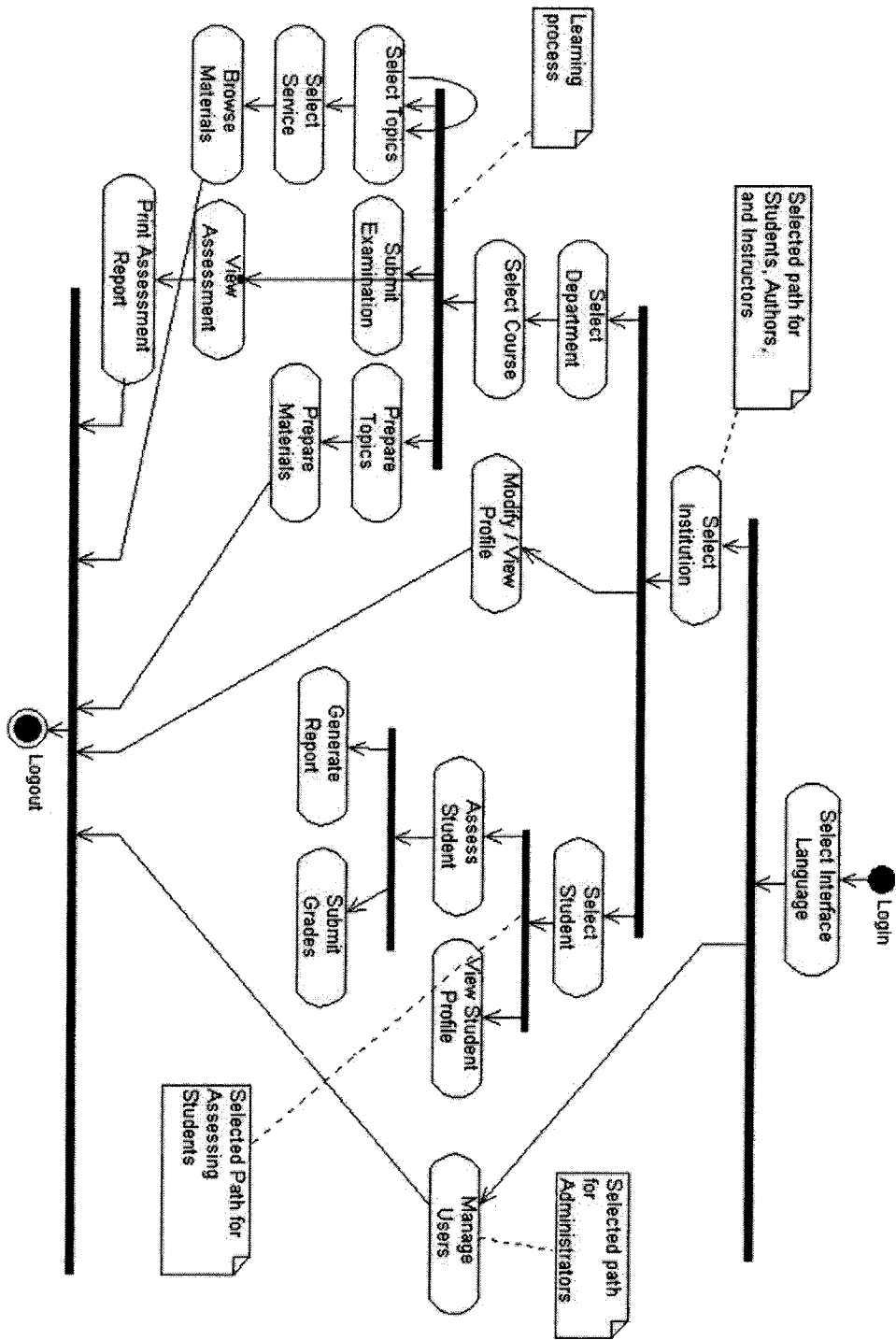
#### 3.5.1.2 Performance issues

The system is expecting to respond efficiently under all circumstances known regarding the Internet. One exception to issue is related to the collaboration interaction and file transfer. For both of these cases, one can overcome them through the use of higher connections speed. But in case the regular modem speed is the only available option, such services might perform unexpectedly according to the network bottlenecks and connection speed. But in overall, and due to advances in technology, one can always overcome these difficulties. For example, with respect to file transfer, downloading might be solved in a moderately low cost through the use of a download satellite subscription. But still, on the other hand, there is the problem of uploading. So as you can see solutions regarding speed issues are still waiting for technological advances to be solved.

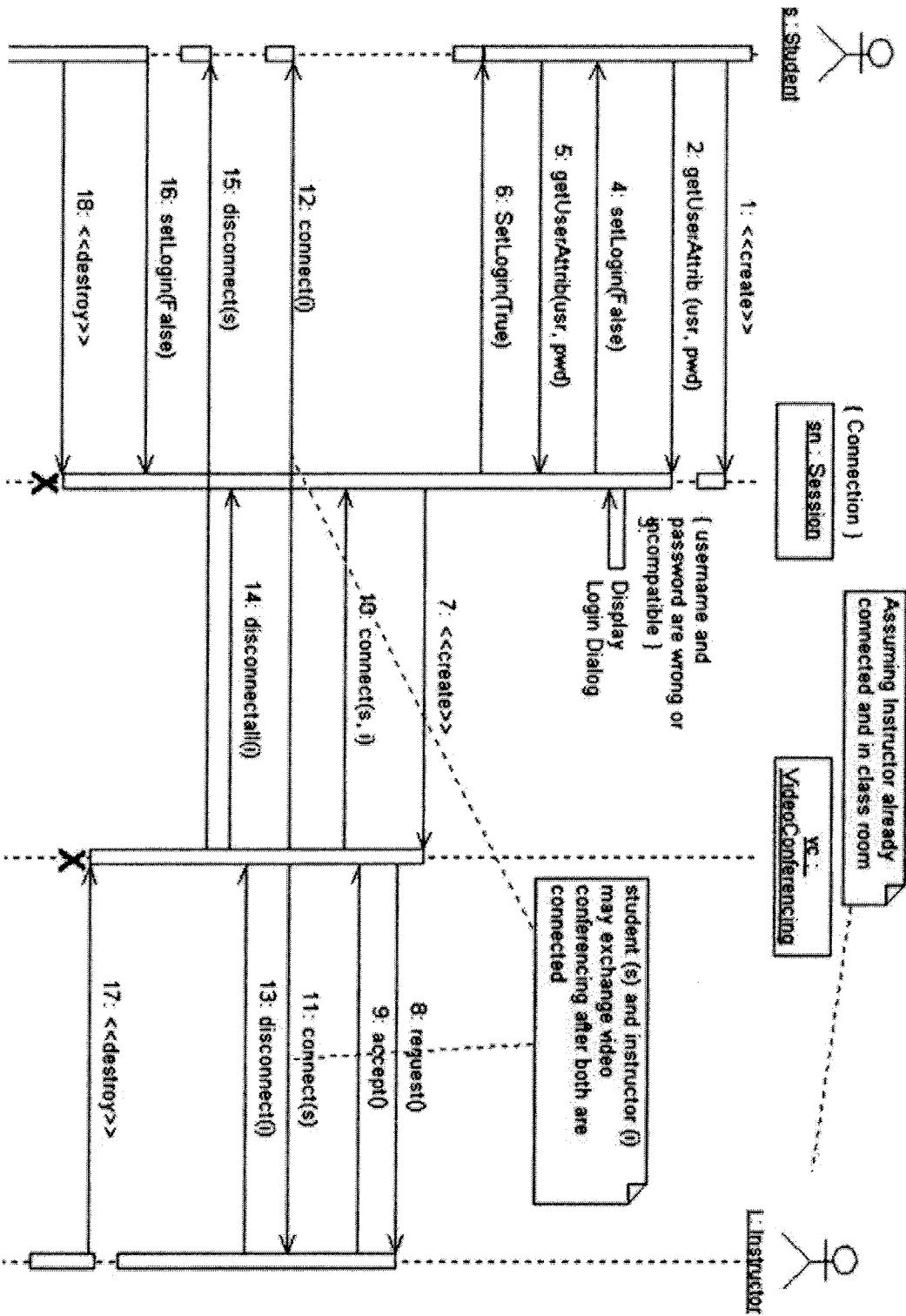
### 3.6 AFDL behavioral model and description

The behavior of the software is presented in this section using UML diagrams.

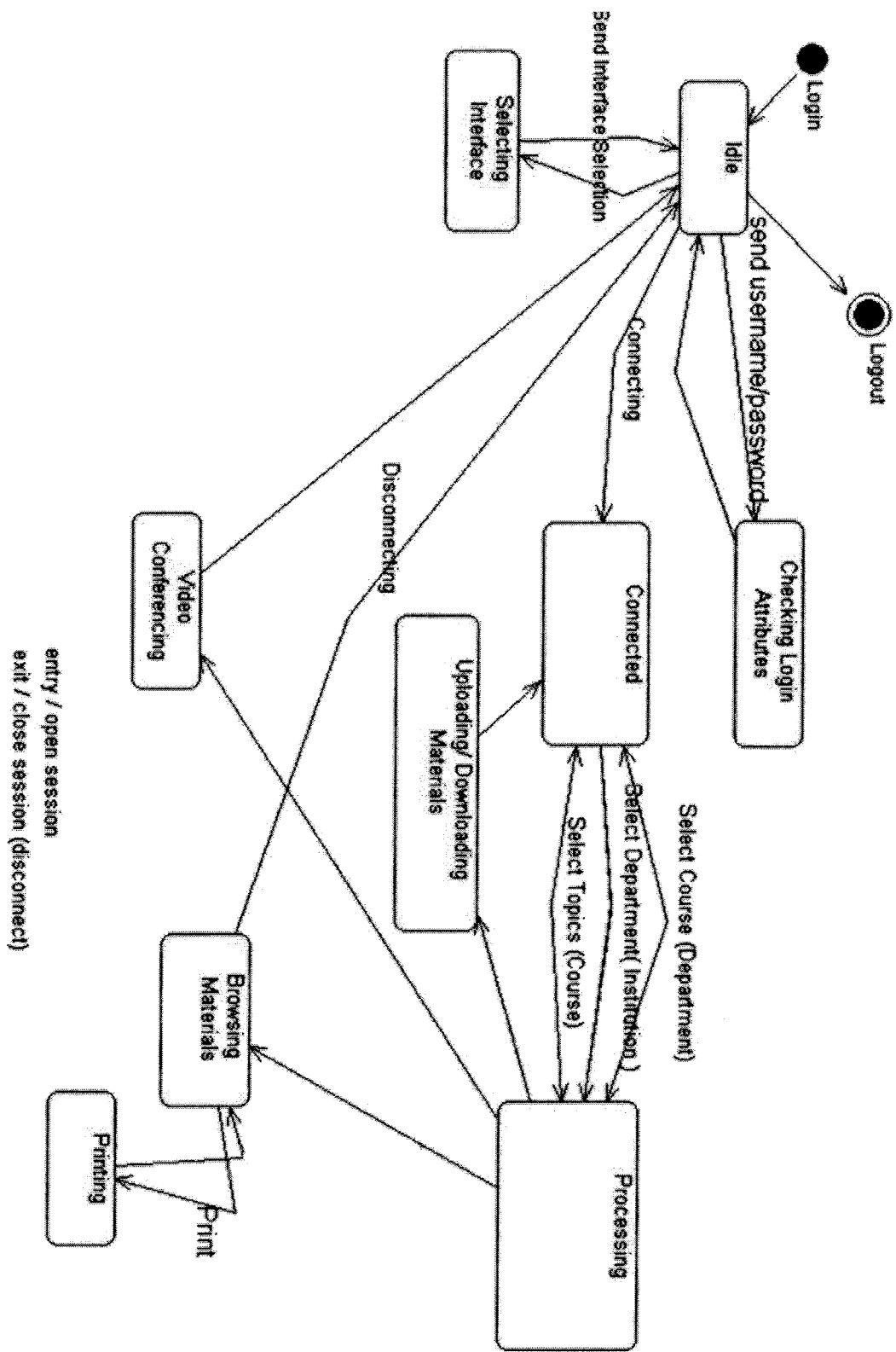
### 3.6.1 Activity diagram



### 3.6.2 Sequence diagram



### 3.6.3 State-chart diagram



(Ref. No. 18, 19, 20, 21, 23, 24: [Crespo-Fontoura-Lucena 1998], [Lucena – Fuks - Milidui' – Macedo – Santos – Laufer – Ribeiro – Fontoura – Noya – Crespo – Torres – Daflon – Lukowiecki 1998], [Vaupé-Sommer 1997], [Alencar - Cowan - Crespo - Fontoura - Lucena 1998], [Fontoura - Pree - Rumpe 2000], [Fontoura-Moura-Crespo-Lucena 1998])

## Chapter 4

### AFDL - IMPLEMENTATION

In this chapter we present the implementation part of AFDL.

#### 4.1 The five-module framework for internet application development

Most current web application development uses a three-layer approach. The three layers are the presentation layer, the business logic layer, and the system layer. The presentation layer is responsible for presenting the content with a user-friendly interface. This is typically accomplished with application development software such as Visual Basic, Visual C++, or Delphi. The business logic layer implements most of the business logic and provides services to the user interface (UI) components in the presentation layer. The system layer provides the network, operating system, and persistent storage for data such as MS-SQL Server. In this section we address the three-layer architecture and prove the importance of using the five-module architecture as presented by: Ebner, Shao, and Tsai. According to them, the three-layer architecture has proven useful in Internet applications. However, it has the following limitations:

- **A system implemented using this architecture has a close connection between the presentation and the business logic.** Usually a static binding between the presentation and the business logic exists because the presentation calls the functions contained in the business logic layer. If there is a change in the presentation or in the business logic, then the parts of the application that reference them must also change.
- **The business logic and system layers are tightly coupled.** Therefore, changing the platform, operating system, or database management system may cause many changes in the business logic and sometimes even in the presentation layer.

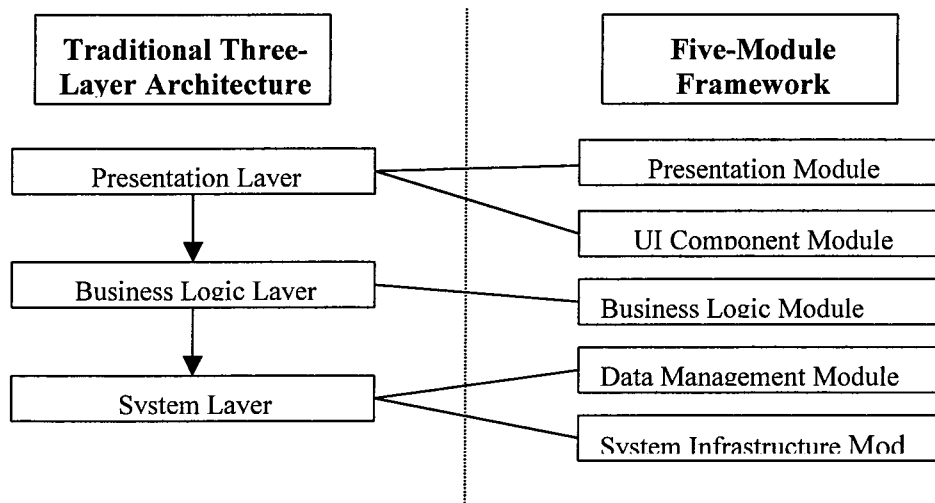


- **In the presentation layer, layout information, navigation information, and UI components are sometimes mixed together.** There are different types of data that are handled by different experts (domain experts, media experts, and programmers). The architecture should reflect these differences.

To address these limitations, Ebner, Shao, and Tsai proposed a five-module architecture: presentation, UI components, business logic, data management and system infrastructure (see **Figure 4.1-1** below). The five-module architecture explicitly separates the data model from the data source. In this manner, it is possible to change the underlying database management system with minimum effort. In addition, the five-module architecture uses a broker interface between the modules. This Interface technology allows developers to add, delete, and modify components with minimum effort. By definition, the layered approach only allows a layer to communicate with the layers directly above and below it. Decomposition of the presentation and system layers requires that the business logic communicate with the presentation, UI components, and data management. This violates the layered principle, thus requiring that the architecture be composed of modules and not layers.

### I- Presentation

This module prepares the content data for display on the web browser. It also provides the runtime binding between the UI components and the business logic. This module functions independently of the content of the page. So, it is possible to change the business logic or UI components without changing the presentation module.



**Figure 4.1-1:** The traditional 3-layer and the 5-module framework architecture

## **II- User interface component**

This module provides UI components for display by the presentation module. It is possible to add UI components without changing the presentation. The UI components convey information to the user through the presentation layer and also accept user input through the presentation layer. Typical UI components are text images, text fields, radio buttons, video, hyperlinks, scroll bars, and tables.

## **III- Business logic**

This module provides the services that the user can request from the application by interaction with the UI components in the presentation module.

## **IV- Data management**

This module provides persistent data storage and management for the application independent of the underlying database. This module provides an abstract interface to the database and a database adapter that allows the system to use databases from various vendors. The data management layer can store and retrieve code and data. All portions of the architecture that use the services of the data management are stored within the data management along with the data that they generate.

## **V- System infrastructure**

This module provides configuration and maintenance of the underlying hardware, operating system, and network.

In an application built using the five-module architecture, the presentation module loads the specification for a page from the database through the data management module. The presentation module creates the HTML for the layout of the page from the specifications. Then the business logic object associated with each UI component is invoked to allow initialization of the UI component. Then the UI component creates the HTML for its display in the browser. Then the page is assembled and transmitted to the user. When the user generates a response to the page. The response passes to the business logic objects associated with the UI components. Then the response passes to the business logic object associated with the entire page. Then the data management layer saves the response and any data generated by the logic objects to

the database. (Note: that a reference to the page and the user who generated the response may be also saved if required.)

The architecture allows the developers to build many applications by binding standard code (the business logic) to standard components. Then if the developer needs custom behavior he or she adds new business logic objects and UI components to the data management layer and binds those to the page or UI components within the page. This allows the capabilities of the system to change without changes to the underlying application or architecture [Ebner-Shao-Tsai2000]).

## **4.2 AFDL interface description**

The software interface(s) to the outside world are described in this section.

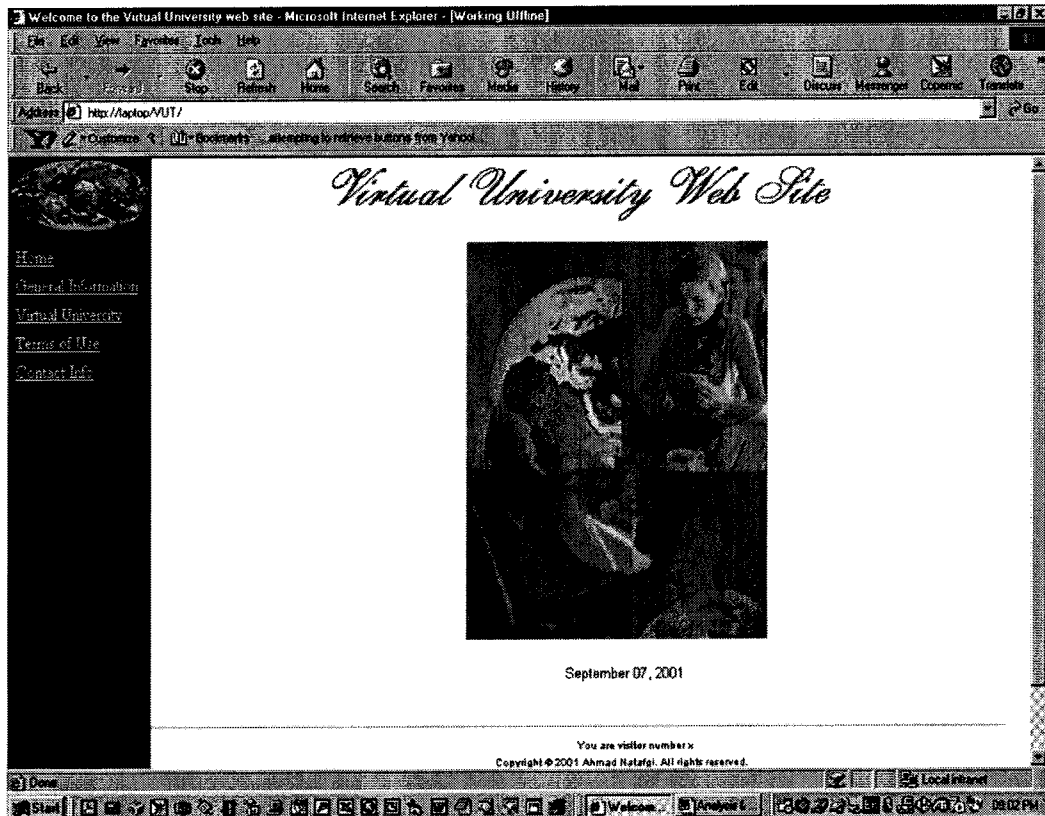
### **4.2.1 External machine interfaces**

Interfacing with other computers is done in case of distribution of the processing power over several computers in order to distribute the traffic reflected on a specific server. It is simply a matter of architecture decision. (For further details, refer to the system architecture section)

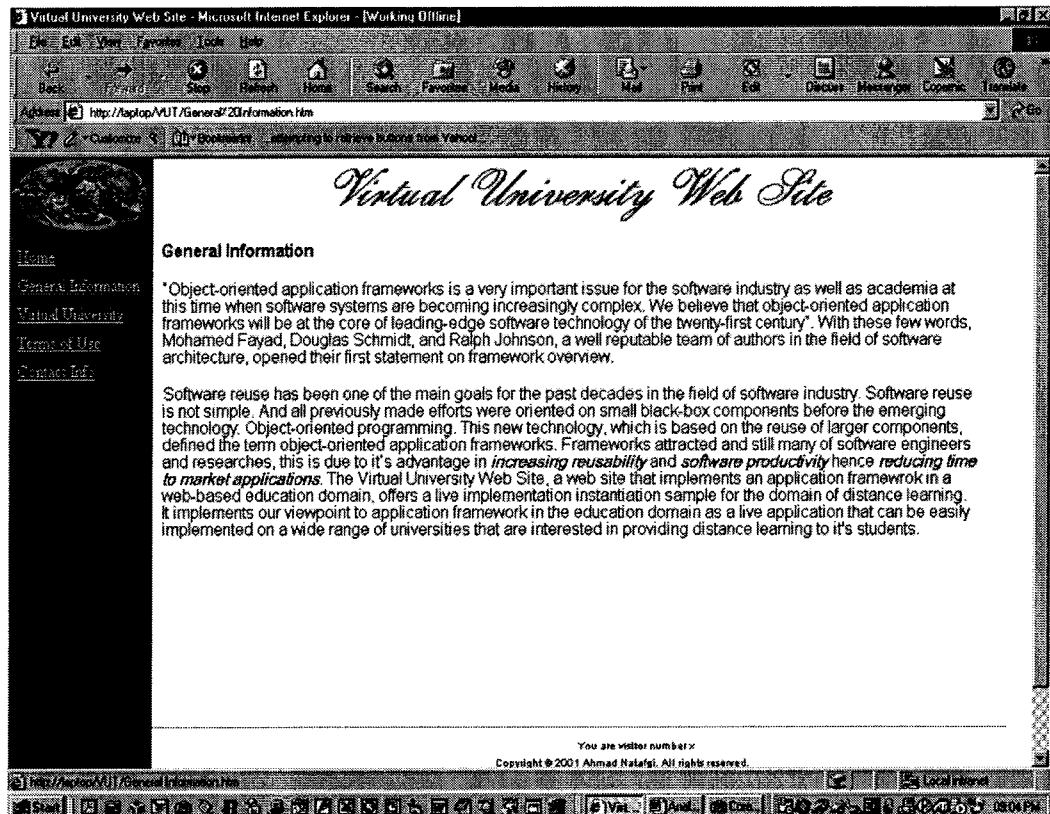
### **4.2.2 Human interface**

An overview of human interfaces designed for the AFDL are presented.

#### 4.2.2.1 The welcome page

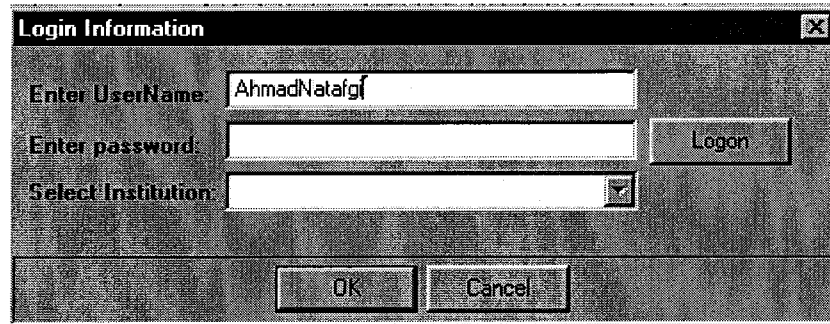


#### 4.2.2.2 The general information page



#### 4.2.2.3 The virtual university page / AFDL login

In this interface the user is prompted to login to the system and to select the institution to login.



**Login Information**

Enter UserName: AhmadNataf

Enter password:

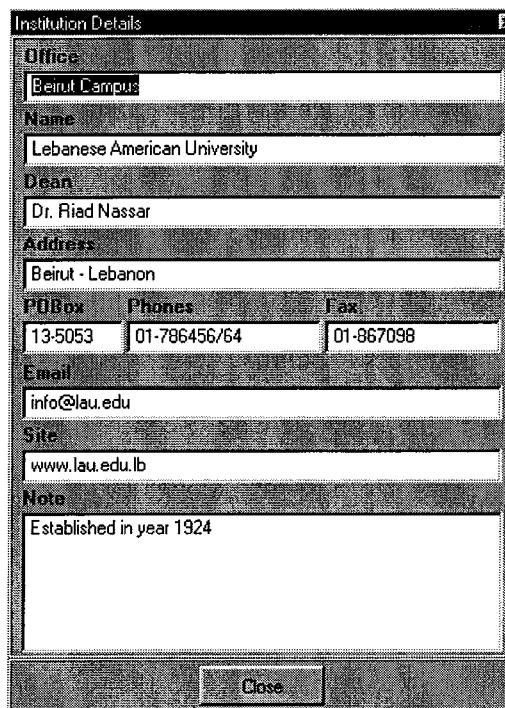
Select Institution:

Logon

OK Cancel

#### 4.2.2.4 The virtual university page / AFDL institution details

In this interface the user browse the details of the institution.



**Institution Details**

**Office**  
Beirut Campus

**Name**  
Lebanese American University

**Dean**  
Dr. Riad Nassar

**Address**  
Beirut - Lebanon

POBox	Phones	Fax
13-5053	01-786456/64	01-867098

**Email**  
info@lau.edu

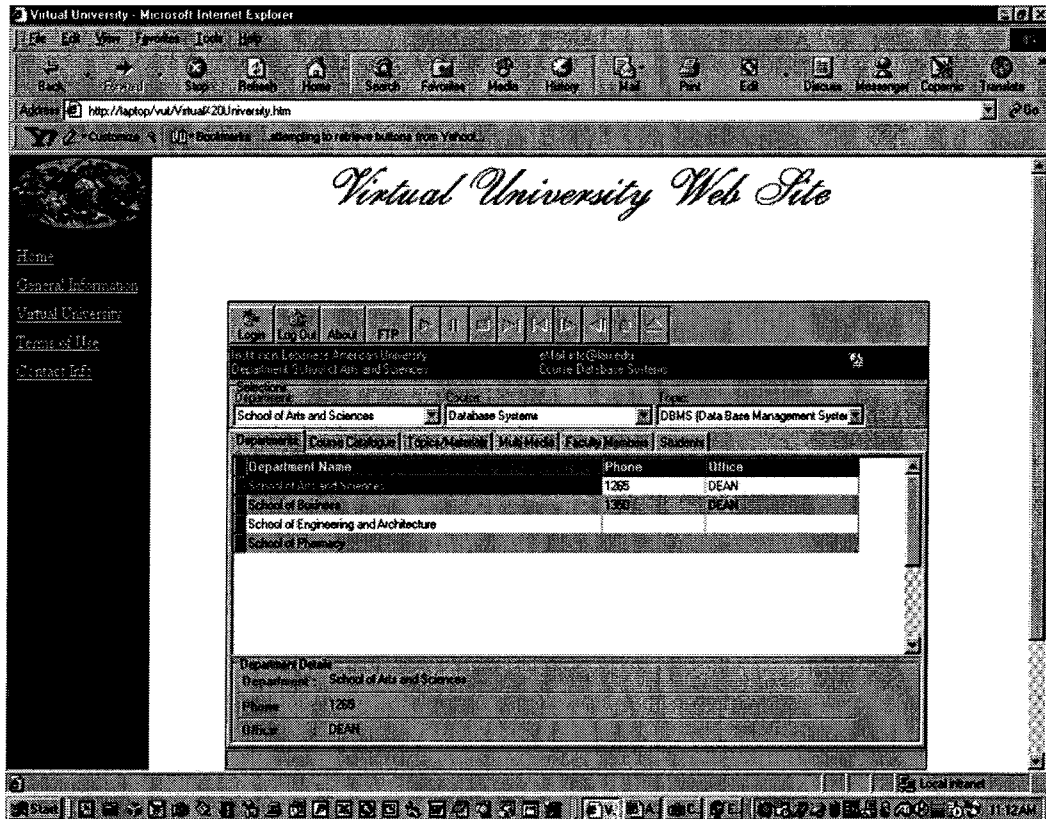
**Site**  
www.lau.edu.lb

**Note**  
Established in year 1924

Close

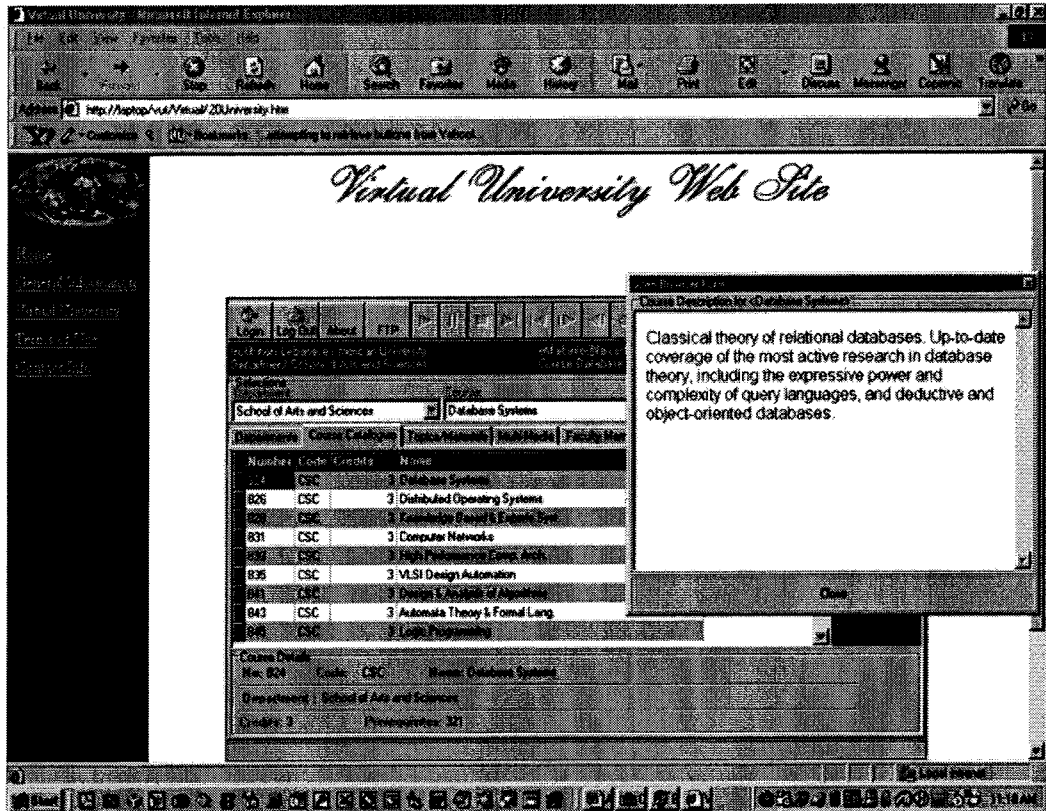
#### 4.2.2.5 The virtual university page / AFDL department browsing

In this interface the user can select the departments related to the institution. And also he can view the institution details and modify it. (According to the access rights).



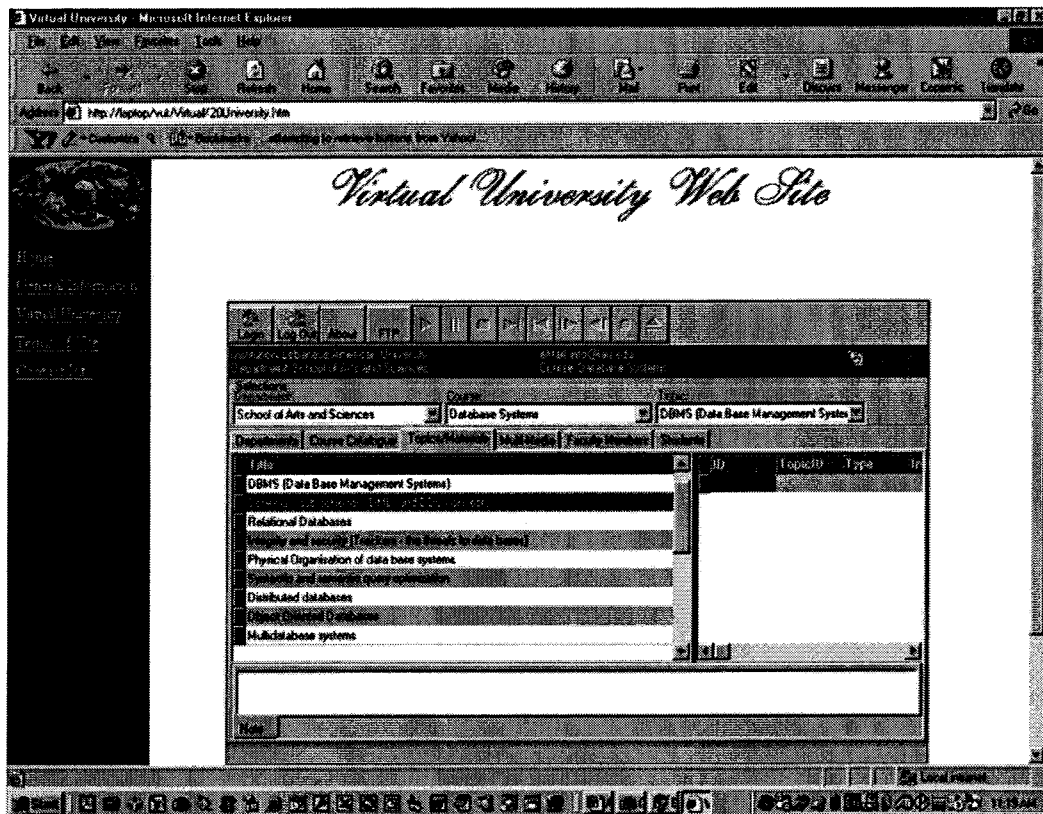
#### 4.2.2.6 The virtual university page / AFDL course catalogue

In this interface the user can browse the courses available and see all course details and prerequisites.

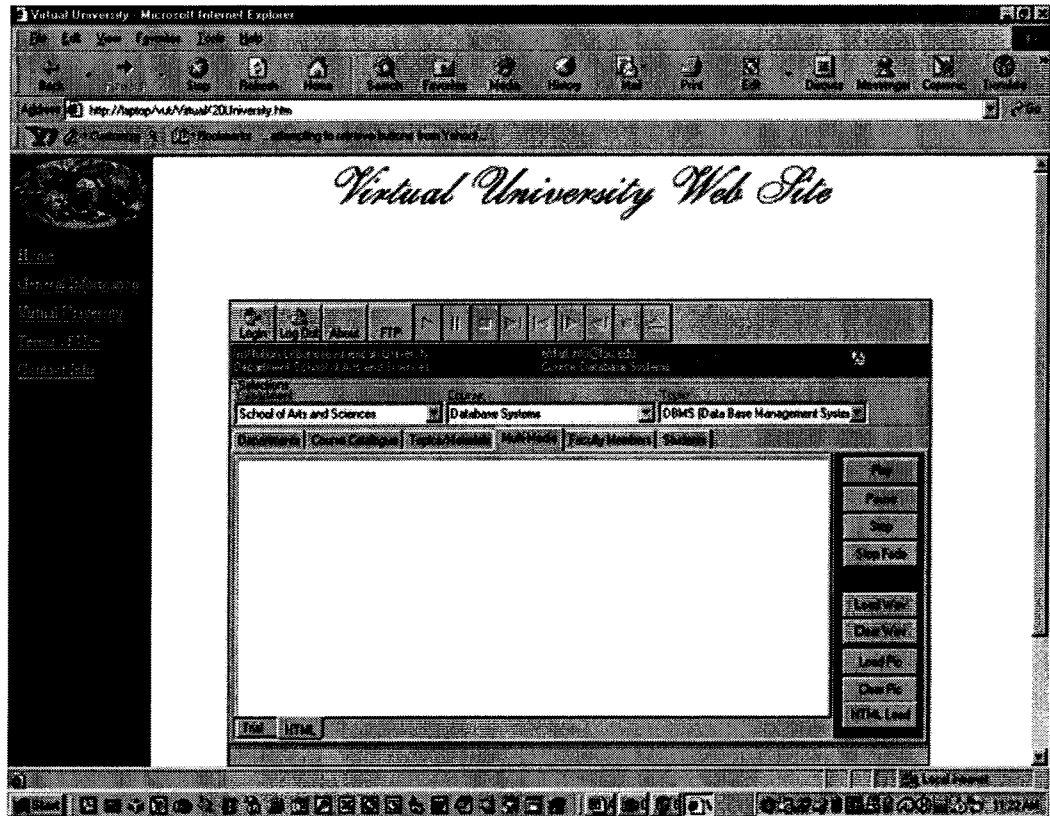




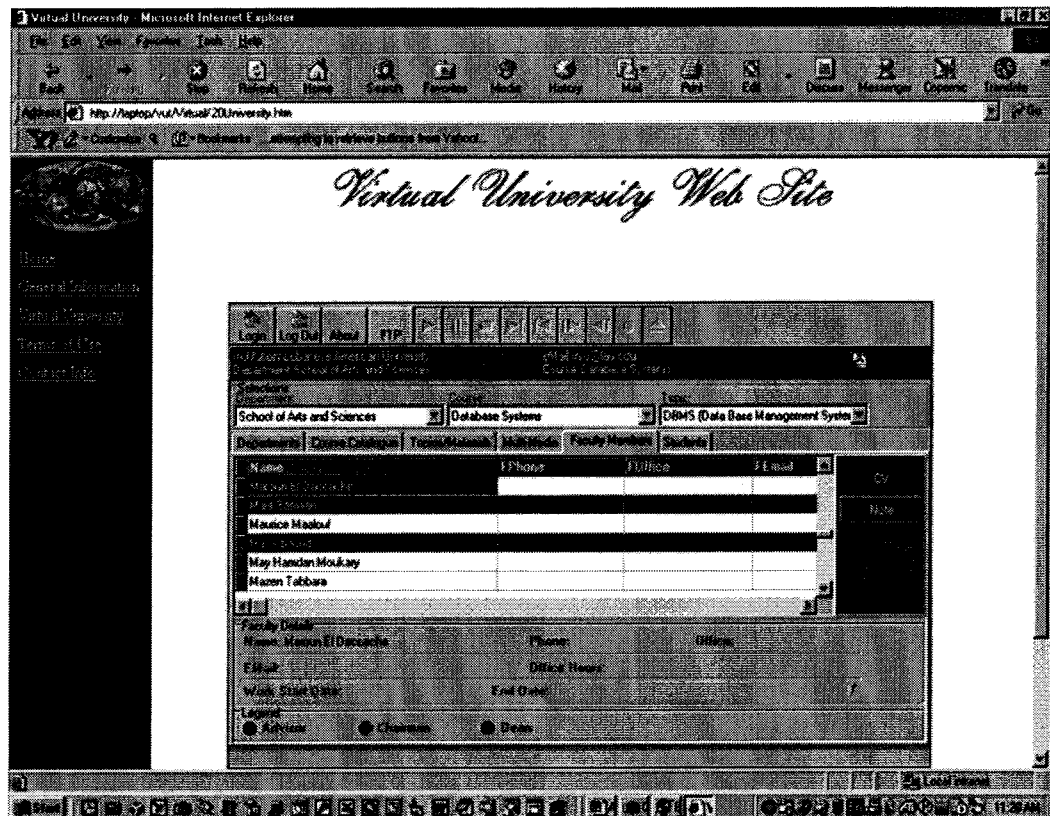
**4.2.2.7 The virtual university page / AFDL topics/materials:** In this interface the user can browse the Topics / Materials for the selected course. (This page is the most important page actually it is the core objective of the system in all).



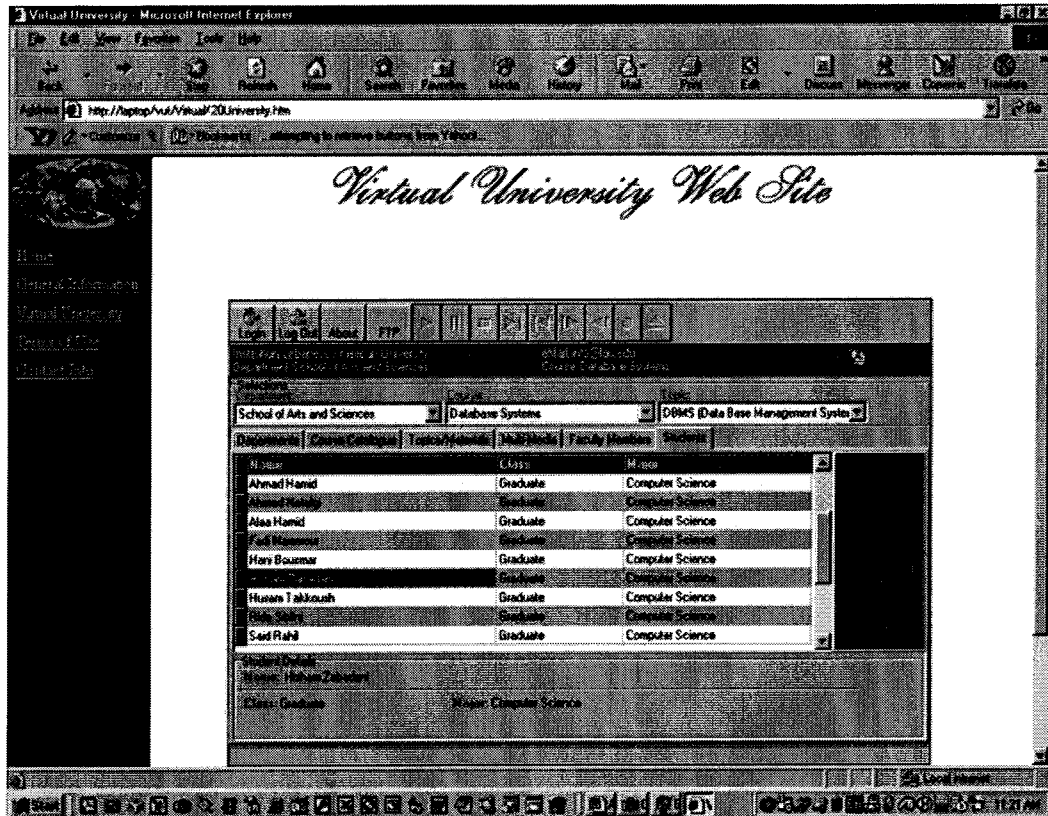
4.2.2.8 The virtual university page / AFDL multimedia: In this interface the user can interact with the system using it's collaboration interaction features.



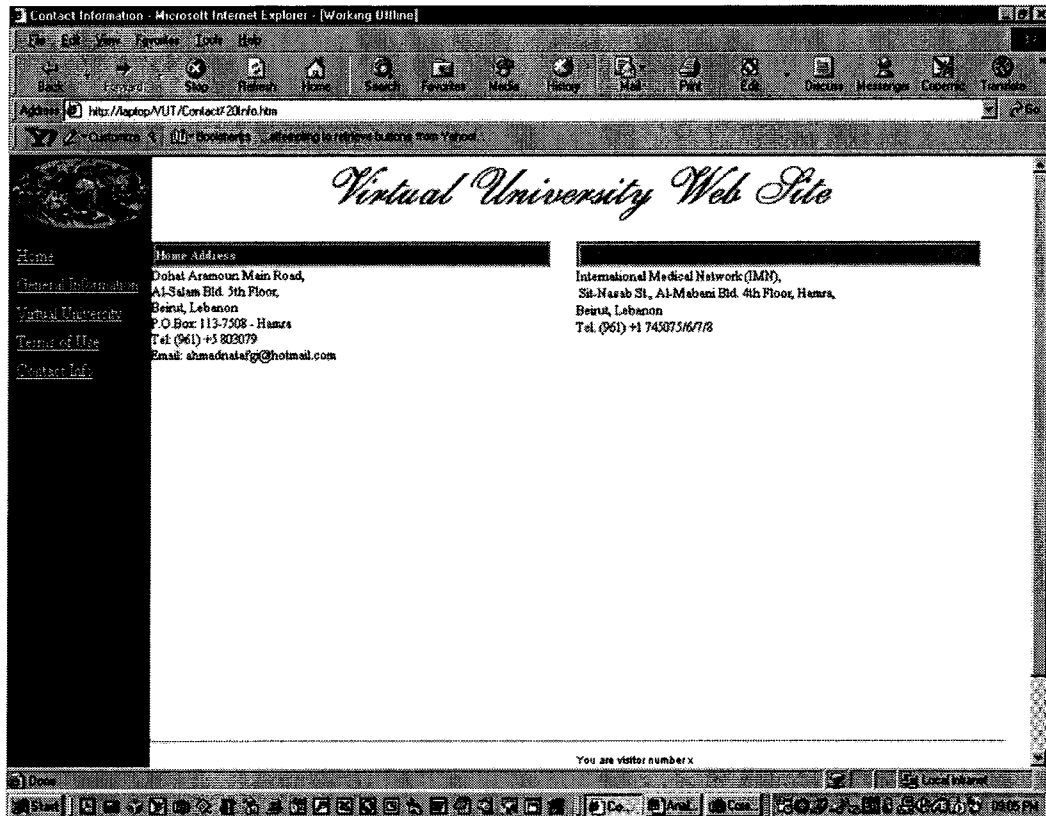
**4.2.2.9 The virtual university page / AFDL faculty members:** In this interface the instructors and students can browse the faculty members information according to access right defined to each user group.



4.2.2.10 The virtual university page / AFDL students: In this interface all user members according to security rights, of course, browses student information.



#### 4.2.2.11 The contact information page



### 4.3 AFDL restrictions, limitations, and constraints

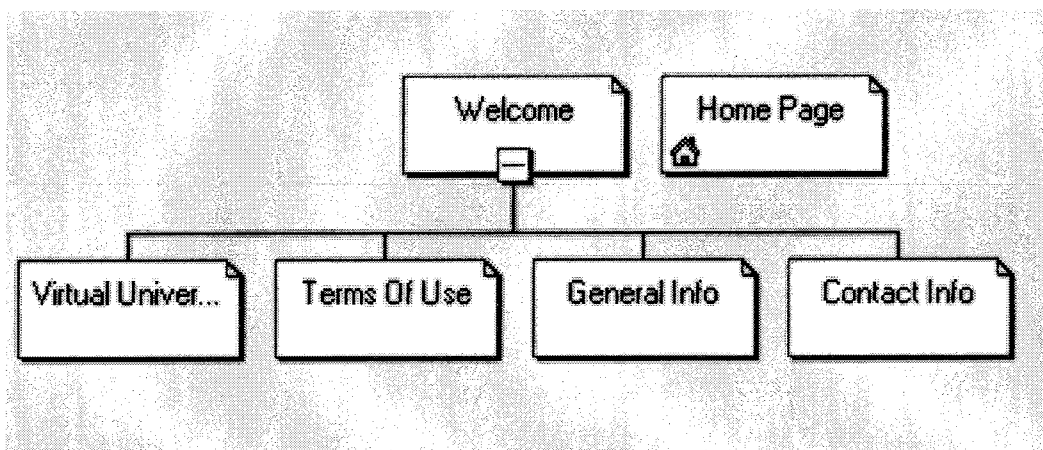
As we have already mentioned in the major constraints section, AFDL's main implementation constraint is related to the constraints inflicted by the world wide web (or Internet) itself regarding the speed issues of communication and all its related problems such as file transfer speed (either uploading or downloading). But in case the framework is to be used (implemented) in high-speed connection environments such as an ISDN or higher connection speeds (128K and above), this constraint would cease to exist. Moreover, even with the existence of such an obstacle and with the emergence of new effective compression algorithms such as the ones used in mp3 file formats and in those used in the collaboration (audio/visual communication) area, we are able to use them effectively to provide the user with the best technology possible in the most cost-effective way without imposing on him additional cost for subscribing in a high connectivity ISP. Of course this constraint issue is reflected on

the time delay imposed on the end-user or student while requesting file transfer service or online collaboration session.

In addition to that, one additional constraint may be imposed by the fact that our **AFDL** is a framework not the final software. Which bares both benefits and constraints at the same time. Regarding the constraints, well, a framework is an unfinished work. I.e. the Hot-spots are still waiting to be implemented usually by experienced developers. Also, these experienced developers need to pass through our documentation and framework components and get to know them before they are able to fully benefit from the instantiation process.

#### 4.4 AFDL site map (or navigation tree)

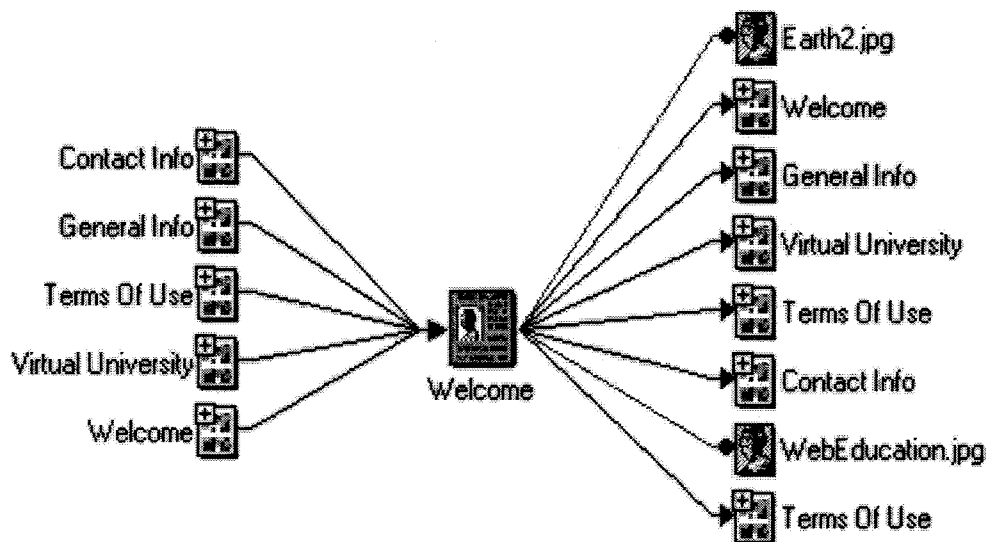
The site navigation tree as already implemented using MS-Front Page version 2000 is as follows:



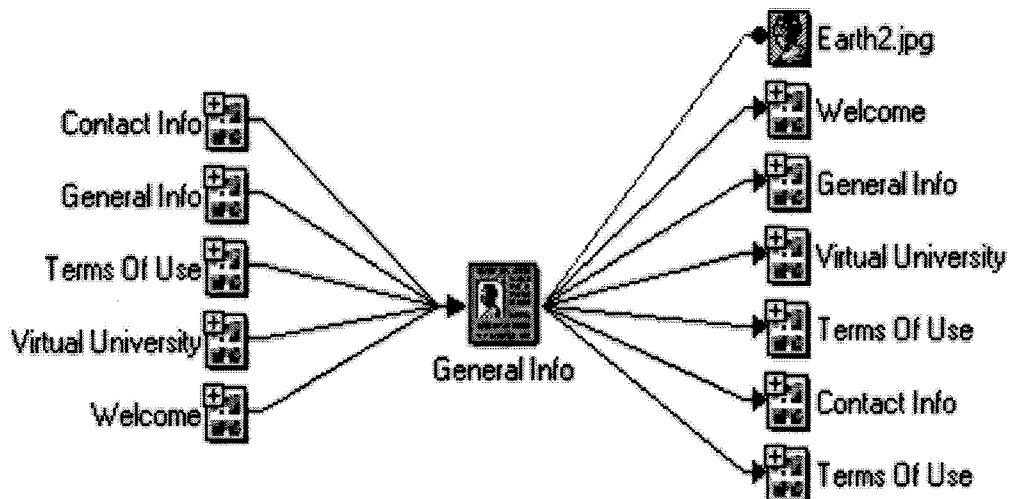
## 4.5 AFDL hyperlinks

The site Hyperlinks tree as already implemented using MS-Front Page version 2000 is presented bellow according to each site page available.

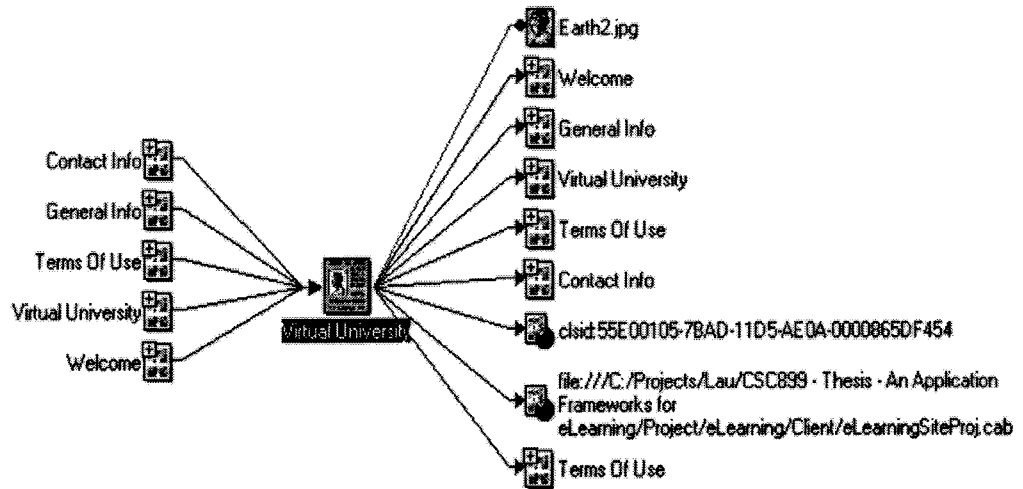
### 4.5.1 Hyperlinks for the welcome page



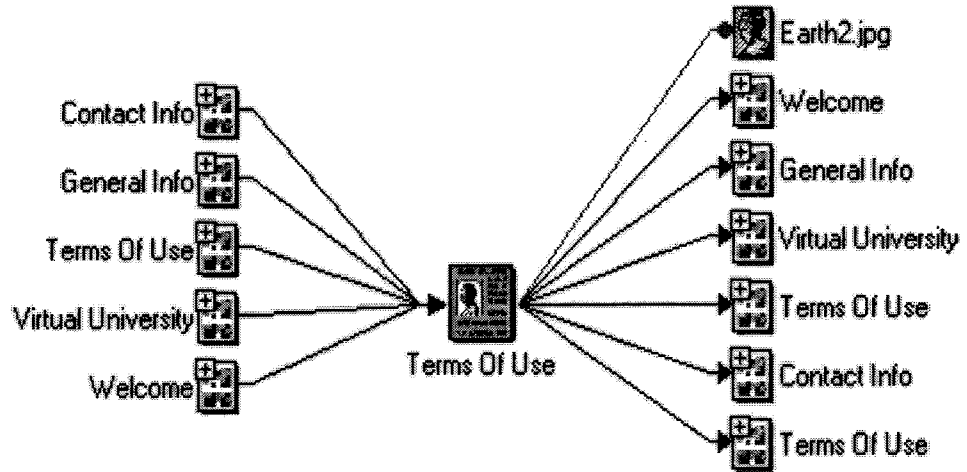
### 4.5.2 Hyperlinks for the general information page



### 4.5.3 Hyperlinks for the virtual university page

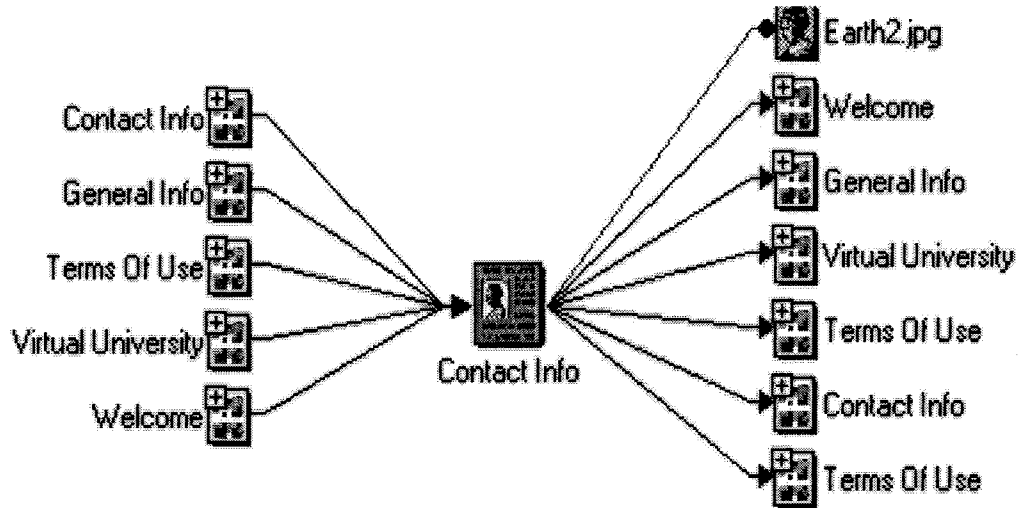


### 4.5.4 Hyperlinks for the terms of use page





#### 4.5.5 Hyperlinks for the contact info page



### 4.6 AFDL architecture

#### 4.6.1 AFDL architecture

The architecture of AFDL is a Web-based architecture. The environment interface (presentation layer) is being developed using Hyper Text Manipulation Language (HTML), which implements most of the site's interfaces. The user interface module is implemented as a one module, an ActiveX component. It is being implemented using Borland's Delphi ActiveX technology. ActiveX technology provides its application developers with the ability to develop advance applications including Web-based applications using any language they are familiar with. For example, Visual Basic developers can implement their Visual Basic version of ActiveX and embed it in AFDL. (Same for Visual C++, Visual Java, and any object-oriented development environment that provides ActiveX technology). Regarding this point, the users (application developers) of AFDL, can include the ActiveX control version of AFDL into any Web-browser (Internet Explorer or Netscape Navigator) as an OLE object (Object Linking and Embedding). For example see the following code:

```

<OBJECT
    classid="clsid:55E00105-7BAD-11D5-AE0A-0000865DF454"
    codebase="C:/Projects/Lau/CSC899 - Thesis - An Application Frameworks
for eLearning/Project/eLearning/Client/eLearningSiteProj.cab#version=1,0,0,0"
    width=690
    hspace=0
    vspace=0
>
</OBJECT>

```

In the above example, we can view part of the HTML code for embedding an ActiveX control into one of our Web pages. Notice that our code for embedding our AFDL ActiveX object is placed inside the opening tag <Object>, and the closing tag </Object>. The command "classid" is assigned a value, which is the ActiveX Identification number provided by the development environment (Borland Delphi in our case). It is provided upon registration. The command "codebase" is used to define the location where the component is actually present (physically on the Hard disk) on the publishing machine. (Usually the ActiveX control is downloaded once to the client's machine upon demand i.e. when the page including the control is requested). Once the ActiveX control is downloaded to the clients machine, it is available for use upon request at any time until a new version is published, where in that case, it is downloaded again and replacing the older version. Note that: by the word registration we mean, that we are registering the control in the operating system (Windows in our case) in order to be available for use later. To register the AFDL ActiveX component (called "eLearningSiteProj.ocx"), at the command prompt, we enter the following: "regsvr32.exe <Physical location>\eLearningSiteProj.ocx" and then press enter. By the <Physical Location> we mean the physical path of the "ocx" file of our AFDL ActiveX component, which we choose to call it: "eLearningSiteProj.ocx".

Next, In the Business Logic module, we see that all of the business logic and rules are being implemented and present inside the RDBMS (Relational Database Management System) we selected (MS SQL Server version 7.0 in our case). The business logic is to implemented as a series of stored procedures and triggers to be defined according to the business logic related to the specific business (industry) and they are to be considered as one of the major hotspots (flexible areas) to be defined later by the developers instantiating the AFDL. In addition to that, general-purpose stored

procedures can be predefined as part of the framework to serve as a general – purpose procedure to be called by the users (application developers) of the framework later. An example of such a stored procedures could be to define a stored procedure for providing data access to any general made reports. (In our case such of these reports are designed using Seagate Crystal Reports and linked to the framework using the shipped-with component to link the output “.rpt” file to the framework from within the ActiveX component).

In the fourth module, the Data Management module, we use the RDBMS: MS-SQL Server version 7.0 to implement our database for the distance-learning domain. It is completely designed to be instantiated automatically with an automatic built in database generation scripts that generates the database dictionary in the Transact SQL language, which is fully compatible with MS-SQL Server (versions 6.5 and above). Including an initialization data for the database. (This module and all of its included scripts are defined in details in the following Chapter).

Finally, in the infrastructure module, we use the IIS – Internet Information Server together with the TCP/IP protocol to communicate with our AFDL site defined under the name VUT (Virtual University Technology). The IIS is used to serve as a dedicated server that publish the contents of the site with all of it’s related materials, web-site pages, database engine (SQL Server), and persistent data. And the operating system used that provides this system infrastructure is implemented using Microsoft Windows NT technology Server. See **Figure 4.6.1-2** below:

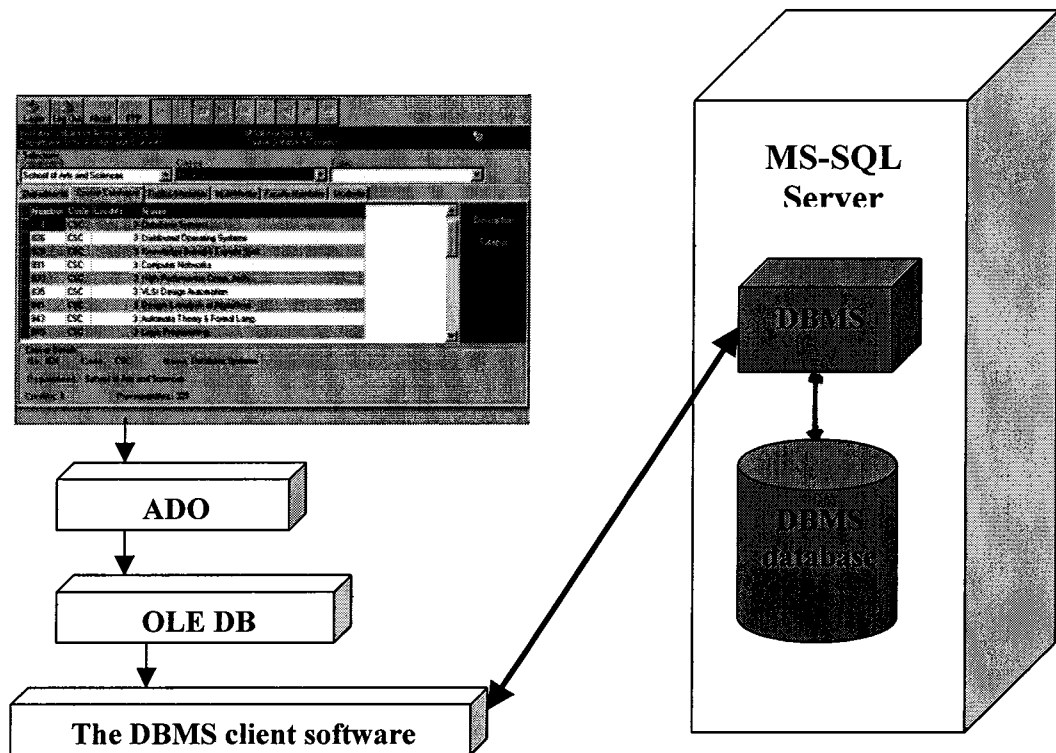


Figure 4.6.1-2: AFDL architecture.

All of the objects manipulated by the environment, like for example courses, students, institutions, departments, faculty, materials, and topics, are stored in a relational database, namely using Microsoft's SQL Server newest technology that is responsible for the persistence of these objects.

In Figure 4.6.1-3 bellow, we present the five-module **AFDL** architecture we have implemented.

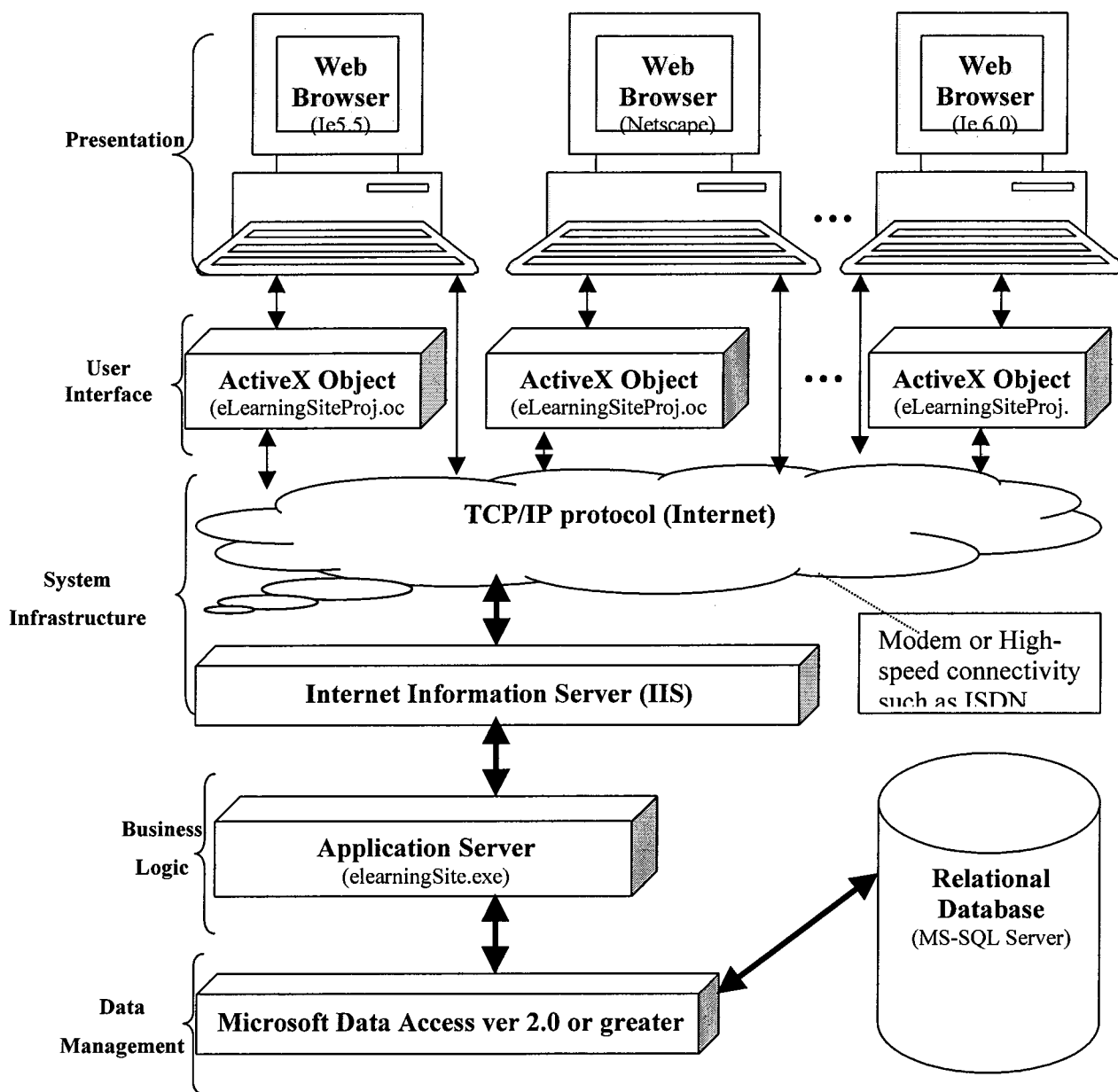


Figure 4.6.1-3: The Five-Module Architecture for AFDL

## 4.7 AFDL and the observer pattern

In this section we discuss the Observer pattern and how it is used in **AFDL**. First we start by discussing the intent behind using an Observer pattern. The Observer pattern defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically. (Also known as Dependents, Publish Subscribe). In order to consider the motivation behind using an Observer pattern, one has to consider the following points:

- Partitioning a system into a collection of cooperating classes leads to consistency problems between related objects.
- Achieving consistency by making classes tightly coupled to each other's is a bad design, because this will reduce reusability.
- GUI (graphical user interface) toolkits separate the presentational aspects of the user interface from the underlying application data. Or, they can work together. Ex. both a spreadsheet object and bar chart object can depict information in the same application data object using different presentations

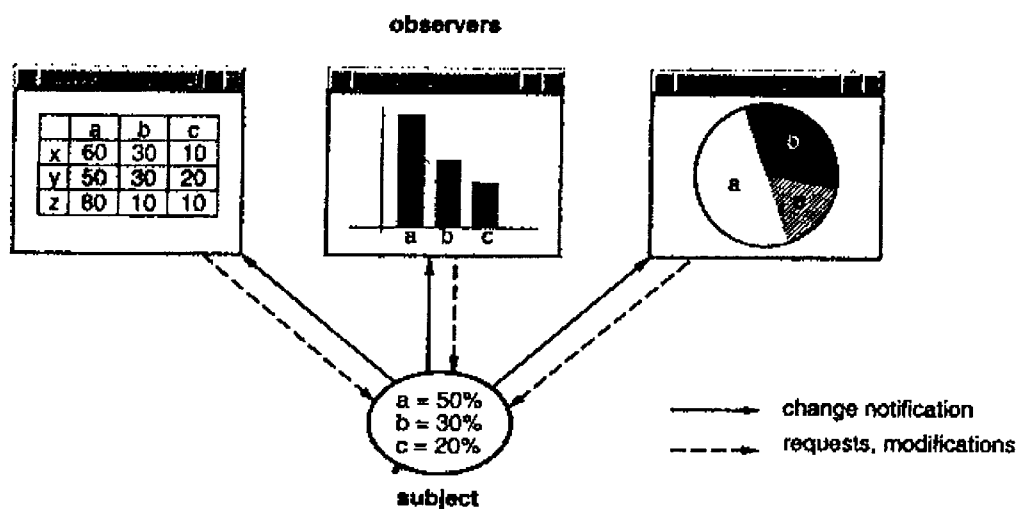


Figure 4.7-4: Observer pattern

In Figure 4.7-4 above, when the user changes the information in the spreadsheet, the bar chart reflects the changes immediately, and vice versa. The spreadsheet and bar

chart are dependent on the data object and therefore should be notified of any change in its state. The key objects in the Observer pattern are *subject* and *observer*. Where a subject may have any number of dependent observers and all of the observers are notified whenever the subject undergoes a change in state. In response, each observer will query the subject to synchronize its state with the subject's state. This kind of interaction is also known as publish-subscribe. The subject is the publisher of notifications. It sends out these notifications without having to know who its observers are. Any number of observers can subscribe to receive notifications. [Gamma-Johnson-Vlissides1995]



Figure 4.7-5: Observer pattern collaboration diagram for AFDL

In Figure 4.7-5 above, we see that when the course table in the AFDL is changed, then all of the related user interface components are being notified of the change and updated automatically. This will prohibit problems from occurring in case of selecting a data in one view would not be reflected in other related views.

## Chapter 5

### AFDL - INSTANTIATION

In this chapter we provide a detailed description of the instantiation phase. We start by giving a brief definition of this step, then describe in details the instantiation phases for AFDL.

#### **5.1 Frameworks instantiation a brief definition and description**

Framework instantiation can be a complex process. It could be far more complicated than just plugging in the components into the framework hot-spots. This process usually is regarded as an impossible step to be completed by the application developers if it is not well documented. In the real world, a proper instantiation of a framework requires a well documentation. Hence, documenting framework well is considered to be an important step for frameworks success. Other authors in this subject have proposed several approaches to instantiate a framework. Such as, the process-based instantiation and the domain-specific languages approach, and others. [Fontoura1999])

#### **5.2 AFDL instantiation detailed description**

In our work, we consider an approach similar to the domain-specific languages approach which is basically a domain-specific language (DSL) written in Transact SQL. This approach is used in generating our domain database and initialization data itself. By DSL, it is meant that, the instantiation code is written to implement a domain specific framework namely in the field of distance learning and the actual language that supports our code is a universally accepted language which is the Transact SQL. So we are using Transact SQL (a universal language) to serve our domain-specific requirements. And hence it is a combination of both. This would be preferable because the application developers are expected to be familiar with



Transact SQL universal language (especially the experienced ones), and so we are to benefit more from the time to be saved instead of wasting it on learning the domain-specific language. To conclude this point, we are using the universal language the Transact SQL and at the same time, we are tailoring this language features to serve our demands to the best suitable manner regarding our domain area, the distance-learning domain. Using this tailored DSL, we are able to generate the framework instantiation code, as we will see in later sections from simply executing the “Transact SQL scripts” in the query analyzer utility that is provided as part of the RDBMS of MS-SQL Server.

In addition to that, and regarding the framework itself, written in Borland’s Delphi, the hot-spots are defined to be open for the application developers to add whatever code they see to be fit for their business domain. For we have designed and implemented a complete architecture, with its corresponding presentation, user interface modules, system infrastructure, and database Management modules. So, what is still uncompleted is the business logic module. Which can be considered as an open code or hot-spots and it is up to the application developers to fill. The business logic module, according to our architecture, can be filled out in two ways: First, inside the application server itself, by defining the business logic reflected in the Remote Data module inside the application server. Second, by defining a series of stored procedures and triggers that can be executed to reflect business logic of the specific industry. To clear one thing though, users of the framework are free to use either way in their implementation, although we would advise to use the features available by the RDBMS i.e. to use stored procedures and triggers instead, because according to our experience, this would provide an open margin for later maintenance and modification.

### **5.3 AFDL database generation script**

Bellow we provide a sample code of the database generation script we have implemented for the **AFDL**. This sample code reflects the major and important sections of the **AFDL**. (For a complete list of the database generation script, please refer to Appendix A).



```
        [ID] [int] IDENTITY (1, 1) NOT NULL ,
        [Description] [varchar] (20) NOT NULL ,
        [Type] [tinyint] NOT NULL ,
        [Interface] [tinyint] NOT NULL ,
        [Course] [smallint] NOT NULL
    ) ON [PRIMARY]
GO
```

```
/****** Object: Table [dbo].[ServiceType]   Script Date: 10/17/01 09:59:24 PM
*****/
```

```
CREATE TABLE [dbo].[ServiceType] (
    [ID] [tinyint] IDENTITY (1, 1) NOT NULL ,
    [Description] [varchar] (50) NOT NULL
) ON [PRIMARY]
GO
```

```
/****** Object: Table [dbo].[Topic]   Script Date: 10/17/01 09:59:30 PM *****/
```

```
CREATE TABLE [dbo].[Topic] (
    [ID] [int] IDENTITY (1, 1) NOT NULL ,
    [CourseID] [smallint] NOT NULL ,
    [ParentID] [smallint] NOT NULL ,
    [Interface] [tinyint] NOT NULL ,
    [Title] [varchar] (100) NOT NULL ,
    [Description] [text] NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

```
/****** Object: Table [dbo].[Material]   Script Date: 10/17/01 09:59:30 PM *****/
```

```
CREATE TABLE [dbo].[Material] (
    [ID] [smallint] NOT NULL ,
    [TopicID] [int] NOT NULL ,
    [Type] [tinyint] NOT NULL ,
    [Interface] [tinyint] NOT NULL ,
    [Content] [binary] (10) NULL ,
    [Text] [char] (10) NULL
)
```

```
) ON [PRIMARY]
```

```
GO
```

```
/****** Object: Table [dbo].[MaterialType]   Script Date: 10/17/01 09:59:24 PM  
*****/
```

```
CREATE TABLE [dbo].[MaterialType] (  
    [ID] [tinyint] IDENTITY (1, 1) NOT NULL ,  
    [Code] [varchar] (10) NULL ,  
    [Description] [varchar] (30) NOT NULL ,  
    [Extension] [char] (3) NULL
```

```
) ON [PRIMARY]
```

```
GO
```

In the database generation script above, we see a Transact SQL version of the database generation code for generating the data dictionary of the **AFDL** inside the RDBMS MS SQL Server. This code and upon execution in the query analyzer utility that is shipped with MS-SQL Server, a complete data dictionary is being created in under the database name Distance Learning and a set of tables related to **AFDL** are ready for the application developers to use and interact with using **AFDL**. The script called “Database\_Generation.sql” we are talking about, and upon opening it in the query analyzer can be executed by simply running the script using the F5 command or selecting from the main menu: Query → Execute and that’s it. After execution the framework users should be provided with the complete database structure ready to be connected to the **AFDL** from the Web-browser.

## 5.4 AFDL data initialization script

In this section, we provide a sample script for initializing the database data. The initialization script is called “Initialization.sql” and it includes a sample data. And in addition to that, the data initialization script includes the basic data information required in any database. Such as the country table for example, which lists all of the countries in the world and is pretty defined to be fixed.

/\*\*\*\*\* Object: Stored Procedure dbo.Initializations Script Date: 10/15/01  
01:52:51 AM \*\*\*\*\*/

CREATE PROCEDURE [Initializations] AS

---

--Fill the Institution table

Set IDENTITY\_INSERT Institution ON

IF NOT EXISTS (SELECT CName FROM Institution WHERE CName = 'Lebanese  
American University')

INSERT INTO Institution (ID, COffice, CName, Dean, Address, POBOX, Phones,  
Fax, Email, Site, Note)

VALUES (1, 'Beirut Campus', 'Lebanese American University', 'Dr. Riad  
Nassar', 'Beirut - Lebanon', '13-5053', '01-786456/64', '01-867098', 'info@lau.edu',  
'www.lau.edu.lb', 'Established in year 1924')

IF NOT EXISTS (SELECT CName FROM Institution WHERE CName = 'American  
University of Beirut')

INSERT INTO Institution (ID, COffice, CName, Dean, Address, POBOX, Phones,  
Fax, Email, Site, Note)

VALUES (2, 'Beirut Campus', 'American University of Beirut', 'Malcolm H.  
Kerr', 'Beirut - Lebanon', '11-0236', '01-340460', '01-351706',  
'webmaster@aub.edu.lb', 'www.aub.edu.lb', 'Riad El-Solh / Beirut 1107 2020  
Lebanon')

Set IDENTITY\_INSERT Institution OFF

Declare @InstitutionID smallint

SET @InstitutionID = (SELECT [ID] FROM Institution WHERE CName =  
'Lebanese American University')

Print 'Passed: [Institution] table'

In the sample initialization script above, we see sample data filling script to fill two institutions to the database: they are the Lebanese American University and the American University of Beirut both of which are located in Lebanon.

## 5.5 AFDL IIS configuration

In this section we describe how to configure the Internet Information Server. First of all if you refer back to Figure 4.6.1-3: The five-module architecture for AFDL, you notice that the IIS is one of the important modules in our architecture. It is embedded in the system infrastructure module, and considered one of the important parts of this module. The Internet Information Server, serve as a sharing utility for users to access. It shares the Web-site contents and the ActiveX object. Once the IIS configuration are set, users connected to the site are able to browse the contents of what we have called the **Virtual University Web-Site**. The **Virtual University Web-Site** is the web-site contents and implementation part for **AFDL**. (Refer to section 4.2 AFDL Interface Description for more details).

In order to configure the IIS on the server side the following steps must be completed:

1. Create a new directory under the "Default Web Site" folder and give it a name, for the purpose of demonstration we call it: "**Vut**" (short for Virtual University Technology – our naming convention).
2. Under this newly created directory, take the contents of the **AFDL** shipped on the CD-ROM media under the directory "Site" with its sub-directory source, which contains the "jpg" files.
3. Right click on the "Vut" directory and select properties. Under the tab named "Virtual Directory, make sure that Execute Permissions option is set to: "Scripts and Executables" from the combo-box.
4. Under the tab named "Documents", make sure to add the default page called: "Welcome.htm" in order to be your default Web-site page.
5. Finally make sure that the IIS is in the running mode and that's it. Your server should be ready to serve your customers.

## 5.6 AFDL miscellaneous configuration

Other miscellaneous configurations might be required to be set before using the Application Framework for Distance Learning **AFDL**. Such of these settings are to be made onto the registry of the operating system used (Windows operating system – our selected platform). To start with, run the command “regedit” in the command prompt and edit the windows registry to include a new directory name “Distance Learning” under the main directory “Software” under the main registry key named: “HKEY\_LOCAL\_MACHINE”. Now, under the directory “Distance Learning”, create two additional sub directories called “Client” and “Server” respectively. Under the folder: “Client”, define a new string value called: “WebConnection” and assign to it the name of the client machine you are working on. Second, and under the folder: “Server”, five new string values as are to be defined, for example see the list bellow:

1. DBMode = 0
2. DemoDB = “DistanceLearning”
3. Password = “”
4. ServerName = “Laptop”
5. UserName = “sa”

The first string value called: DBMode, it is used to hold the type of database currently under use. (a demonstration database, a production database or others... - defined in the application server side “eLearning\_s.exe”). The second string value, holds the name of the database and is assigned to the name of the database currently found inside the MS-SQL Server. The third string value and the fifth string value, holds the password and username respectively that are defined in the database in the MS-SQL Server for access rights to the database. Finally the forth string value, holds the computer machine name in which the server application side is running onto.

## Chapter 6

### AFDL - MAINTENANCE

In this chapter we provide a detailed description of the maintenance phase for **AFDL**. We start by giving a brief definition of this step in general, then describe in details the maintenance phase for **AFDL** in specific.

#### **6.1 Frameworks maintenance: vendor viewpoint and general issues**

The same characteristics that make frameworks reusable also place a serious support burden on the framework vendors, who must plan their technical and support strategy accordingly. The strategy must take account of the fact that the vendor has very limited control over the framework's use by the customer and cannot expect perfect communication or a perfect meshing of maintenance goals.

Internal and commercial frameworks measure success by the extent of their reuse. But any worthwhile software system evolves over time. New features are added, performance is enhanced, and new software environments and new hardware platforms are supported. In addition, framework architectures will evolve and white-box components will migrate toward black-box implementations. At the same time, new framework users will have new needs that are not met by the current version. This evolutionary process brings with it some problems that must be addressed by both the customer and the vendor. The vendor, in order to remain competitive, must remain involved in customer issues.

Views and needs of the customer and vendor do not match. Such of these differences are:

1. Framework vendors usually misinterpret their customer's needs from the framework, where they think that framework customers care about the framework product itself, but it turns to be that they care only about the final product or end-result application. The customers are usually interested in part of the framework and hence they don't understand the high-level architecture



of it. So a sub-optimal usage for the framework may lead to a frustrated vendor. Where the customer is interested in making his application work, and in a sustainable design, the framework vendor in the other hand is interested in design quality and integrity.

2. Second, problems that occur in the framework can be divided between vendor errors and customer misunderstandings. Actually, all errors are referred to being vendor errors until they are proved to be otherwise. Then, even if the error is the customer's misuse of the system, the framework vendor must still help the customer find a solution.
3. Framework customers are expected to consult vendors extensively in the use of the framework and will provide feedback on the problem areas. Regarding this idea, one should point out that customers will report a major problem but might fail in doing so for minor ones. Hence, this would lead to a gap in the frameworks version. Where the framework customer might correct some minor problems in the framework and for whatever reason fail to report it to the vendor. In this case, the framework, would ship a newer versions of the framework without taking into consideration these few minor problems.

So, in order to eliminate such problems, the framework vendor must be able to proactively learn about the customer's experience with the framework, i.e talking directly with the application developers and designers instead of leaning on the surveys or reports from customer representatives who are not involved in using the framework. Hence the framework vendor must be able to compensate to the less optimal usage of the framework by the customer.

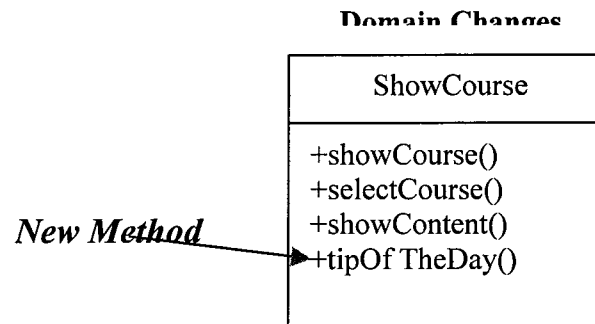
Moreover, framework maintenance or support as viewed by the vendor serve as an important source of revenue in case the framework is for a large system. But in case the framework is considered to be a small-scale framework, then in this case, it is better off for the vendor to lower the support costs as much as possible because the cost of the support is bundled with the price of the framework. In addition to that, it is often the developers are the ones who take over the customer support issues and hence are prevented from doing their assignments, working on future development. Also, some customer might have specific changes that have little general applicability. This poses a problem for both the customer and the vendor.

A solution to these problems would be to provide special contracting. If the requested changes to the framework is large enough and are too expensive to be handled by the designed-in customization mechanisms, it can contract to have the vendor's team make these modifications, which would be implemented in a separate product version. [Fayad-Schmidt-Johnson1999]

## 6.2 AFDL maintenance and specific issues

In this section and regarding **AFDL** maintenance, one might be faced with the case where the **AFDL** does not support the required customization and the application developers might need to violate its structure, a phenomenon referred to as an architectural drift. In this case, the intended framework architecture and the architecture that underlies the current implementation **AFDL** instance would be different. In such case, an already established concept known as the unification rules, can be used to avoid such a phenomenon, by making the necessary transformations in the framework structure to incorporate the changes required by a given framework instance. By the term: *unification*, we indicate that the rules are used to “unify” the modified framework architecture with the architecture of **AFDL**. According to Fontoura's work in this field, there are three unification rules that are to be applied in order to maintain the framework [Fontoura 1999]:

1. Variation method unification: is applied when an instance must change the implementation of a kernel method. After this transformation is applied, the method is defined to be a variation method hot-spot.
2. Extension class unification: is applied when an instance must add new methods to a kernel class. After this transformation is applied the class is defined to be an extension class hot-spot.
3. Extension interface unification: is applied when new class must be added to the framework, and this addition is not supported by any of the framework extension interfaces. After this transformation a new extension interface is created. It can be a new class/interface or it can be one of the classes/interfaces already defined in the framework structure.



**Figure 6.2-1:** Extension class unification

In addition to that, changes to the database structure can be implemented in a similar way. Where addition of new attributes provided they are generic enough might be allowed, modifying an attribute size, or even to add new tables to the database. But bare in mind that modification made to the already present design such as deleting an attribute or changing an attribute data-type would be a highly undesirable step especially deleting an already existing table (schema), which would be in this case prohibited. Because such modifications would definitely impose a chain reaction of changes on already instantiated versions and might impose so many implications that would be difficult to recover from, such as braking a relation between two tables which lead to a chain reaction of braking other relations in it's way.

For example, deleting the course table from the database structure would definitely break the framework into pieces and would be definitely prohibited in such a case. But in case one need for example to enlarge the size of the attribute description in the course table from 100 to 255 (supposedly it was previously 100 char in size), then this would have not empirical changes on AFDL what so ever.

## Chapter 7

### FREQUENTLY ASKED QUESTIONS

In this chapter we state an overview of widely used frameworks, their classifications, their strengths and weaknesses, how to use them, how to learn them, and how to develop them. (Also, we summarize a number of papers done in the field of application frameworks in general and object-oriented enterprise frameworks in specific.)

Most commercially available frameworks seem to be for technical domains such as user interfaces and distribution, and most application-specific frameworks are proprietary. We will give a comparison and contrast between frameworks and other reuse techniques and describe the trade-offs involved in using frameworks and when are they appropriate.

#### 7.1 Pragmatic issues behind framework design

##### 1. When a framework provides an “approximate” fit to an application, how can a user expect to modify or extend the framework?

Our approach to framework design is based on the idea that any framework design can be divided into two parts: The *Kernel* sub-system design and the *Hot-spot* sub-system design. The *kernel* sub-system design is common to all the applications that the framework may generate, and the *Hot-spot* sub-system design describes the different characteristics of each application that can be supported by the framework. The hot-spot sub-system uses the method and the information provided by the kernel sub-system and may extend it.

We use the term *viewpoint* to represent a standard object-oriented design associated with a framework perspective.

Roberts and Johnson state that “Developing reusable frameworks cannot occur by simply sitting down and thinking about the problem domain. No one has the insight to come up with the proper abstractions.” They propose the development of concrete

examples in order to understand the domain. Our strategy is quite similar, analyzing each one of the viewpoints as a concrete example and deriving the final framework from this analysis.

Once all the relevant viewpoints are defined the kernel design structure can be found by analyzing the viewpoints design representations and obtaining a resulting design representation that reflects a structure common to all viewpoints. This common structure is the “unification” of the viewpoints. The common part will compose the kernel design sub-system.

The elements that are not in the kernel are the ones that vary, and depend on the use of the framework. These elements define the framework hot-spots that must be adaptable to each generated application. Each hot-spot represents an aspect of the framework that may have a different implementation for each framework instantiation.

Thus to modify or extend a framework a user is expected to reanalyze what the relevant viewpoints are and redefine the framework kernel.

## **2. How can a framework support product evolution as the fundamental application domain changes?**

If a domain changes, new viewpoints will arise and the framework kernel will also change. However, if the current version of the framework can cope with the new application requirements we propose the use of domain specific languages to generate the new application, which will be a new framework instance.

The use of the DSL allows the designer to develop quickly and effectively a complete software system. The proposed instantiation process uses the following elements: DSLs, the framework, and a transformational system. The DSLs and framework gather the domain main concepts. The transformational system is used to map the specification written in the DSLs to framework instantiation code. A customized application is the combination of the framework with its instantiation code.

**3. What types of problems arise when multiple frameworks must be integrated?  
How do we go about structuring frameworks to ease the difficulty of integration?**

The separation between kernel and hot-spot sub-systems, and the use of domain specific languages to instantiate the framework can facilitate the integration of framework.

**4. In what application contexts is framework development desirable, or even practical?**

There are various application areas that are not established yet and for which new ideas and models are presently under development and evaluation. These are domains in which the possibility of rapidly building new applications is essential and strategic from a practical point of view. Examples of application domains that can be classified in this category include Web-Based Education, electronic commerce, biology, and financial market.

An interesting strategy for developing new applications for these domains is the development of object-oriented application frameworks.

**5. What is the relationship between frameworks and COTS?**

The black-box reuse of COTS can be interesting, however, the reuse and extension of COTS depends highly on the documentation and is very similar to the problem of integrating two frameworks.

If a COTS application is well structured and documented we can easily extend it or even define a new language, which will work as an API, to manipulate it.

[Alencar-Cowan-Crespo-Fontoura-Lucena1998]

## Chapter 8

### CONCLUSION AND FUTURE WORK

In this chapter we present our conclusions based on the work described throughout the work, and lists our contributions in this work with a description of how they have been accomplished. Finally, we suggest directions for future research and new implementation features (ideas) regarding the unfinished work of AFDL.

#### 8.1 Conclusion

“Object-Oriented application frameworks will be at the core of leading-edge software technology in the twenty-first century. ... There are many open research problems associated with better ways to express and develop frameworks. ... As software systems become increasingly complex, object-oriented application frameworks are becoming increasingly important for industry and academia. The extensive focus on application frameworks in the object-oriented community offers software developers an important vehicle for reuse and a means to capture the essence of successful patterns, architectures, components, and programming mechanisms.” (Ref. No. 41: [Fayad-Schmidt-Johnson 1999])

This work presents an Application Framework for the domain of Distance-Learning (AFDL), which can be used by application developers to instantiate a semi ready-made application for the distance-learning industry. AFDL presents to its users, application developers and software engineers, a reusable design with its entire constituent components composed of the adopted *patterns* such as the Observer pattern, *architectures* such as the Five-Module Internet architecture, *components* such as the FTP, Query builder, and some of the *programming mechanisms*. We present the analysis and design for our application framework in the distance-learning domain in addition to the implementation part and its corresponding Web-based architecture. We also discuss the instantiation mechanism and maintenance general issues to be taken into consideration and its related specific issues regarding AFDL. Finally, we provide some of the frequently asked question and answers about frameworks in general.

## 8.2 List of contributions

The major contributions of this work are:

- *AFDL Framework*: This includes the framework itself represented as an ActiveX control, which includes an implementation of the functional specification for the Distance Learning domain. In this step, we provide the user, application developers with an “ocx” object that can be integrated within any object-oriented application development environment such as Visual Basic, Delphi, Visual C++, Visual Java. Of course, AFDL being a Web-based framework itself, the AFDL “ocx” object can be integrated within the Internet explorer to provide to its end users connectivity to the Internet. The importance of this point is viewed from an architectural point of view and in addition to that, some of the patterns such as the Observer pattern have been implemented. Also, the availability of the built-in components is viewed as an additional successful factor to AFDL.
- *UML Diagrams*: A UML representation of the Distance Learning domain. We provide in this work the complete set of static and dynamic diagrams defined using the Universal Modeling Language (UML), being itself a universally acceptable modeling language, which are ready-for-use to the application developers in this field (The distance learning field). The importance of this point is that the application developers can get use of the reusable design and go ahead to implement the specific business logic instead of worrying about the analysis and design phase.
- *Distance Learning Database Structure*: We include in this work a complete set of database structure for the Distance Learning with all of its related schemas and there relationships between each others, not to forget the well-studied attributes for each schema and their potential extension mechanism built-in feature.
- *General Domain-Specific Tools*: These include: File Transfer Protocol tool that handles all file transfers from and to the server providing services to the end-users on the Web, Query builder that helps the end-user to build any form of query to be used later as an input for execution on the database in order to



return result on any query he might need to know with simplicity. The importance of this step is in providing COTS ready for usage, as the black-box components.

- *Database generation script*: An automated database generation script defined in Microsoft SQL Server. In which the database can be automatically generated in any Distance Learning computer server machine. Hence, simplifying the process of AFDL instantiation and implementation. The importance of this feature is to in its automated functionality. Although this feature is just an implementation of the previously mentioned point (*Distance Learning Database Structure*) but it gives the application developers the ability to automatically generate and instantiate the database on any server machine that has MS-SQL server installed on (without user intervention).

### 8.3 Future work

Many ideas and directions for future research and implementation on the AFDL may be proposed. Bellow is just a list of our current proposals:

- *Complete Query Builder*: This is part of our unfinished work, where we have implemented just part of it, the *Free Query* part and still need to implement the builder part of it. The final product is supposed to be embedded in AFDL as a component. (A black-box component).
- *Search Utility*: This utility is supposed to provide users with the capability of search inside the database for any keyword or statement. It may be implemented as an advanced “Find dialog Box”. This utility differs from the above in that it is a user-friendly interface, while in the case of the above, it is for application developers and more advanced users, for example in one case, the user might be required to know in advance the Transact SQL commands, while in regarding the Search utility no such requirements are needed.
- *Filter Utility*: This utility adds to the power of the framework in the way user can filter out data for display and for searching. It can be displayed as an advanced “Filter Dialog Box”, to be implemented as a component.

- *Reporting Utility*: This utility might add to the features of the framework in providing to its users the ability to print reports on output devices such as the laser printer a set of information and studying materials collected from the domain itself. It might be implemented as a reporting component such as the Seagate crystal reports component.
- *Graphical Representation Utility*: Regarding this feature, future work must concentrate on the graphical representation of database statistics sharing with the end-users. We mean by that sharing data statistics with end users graphically. For example, the instructors must assess students in one point or the other and in order to do that a complete chart with percentage of each accomplished job might be displayed graphically to help instructors in doing that. Also, the same graphical representation might be used by the students to evaluate themselves up to the current date.
- *Video and/ Audio Conferencing*: This feature is expected to provide to its end users audio and video conferencing between both parties (actors) in hand. To be in specific, Video and Audio conferencing is expected to provide the full visual and hearing communication between students and instructors.
- *Assessment Services*: For example: Quizzes, Self-Assessment. It is a set of self-assessing quizzes ready made for the students to undertake in order to prepare them selves for the actual quizzes. In addition to that, adaptive tests might be considered.
- *Interface Builder*: Fields Properties (Grid View): Hiding/Showing Fields acc to users' Group (Interface Builder (Auto Form) Interface Customization)
- *Personalization Issues*: Ex. Hiding the instructor's CV or Background
- *Tip of the Day*: This feature provide to its end-user with an online tip to be advised for the different types of users of the framework. Of course, tips are provided to users according to which category the user fall in. For example a user connected to the framework as a student (using a registered student id), would be presented with tips of the day for students only (not for the authors or instructors).
- *Authoring site*: The authoring site is regarded as an important part of the framework and actually the success of the framework might be regarded under potential danger if provided without an authoring site. Where the authoring

site enable the authors or teaching assistants in authoring the materials and hence preparing it to be provided online. (Don't mix between authoring the academia and sharing it as an electronic format on the site's database).

- *Multiple Language Selection:* This feature is also an important feature and easy to implement as well that it is to be provided to the end users of any distance learning application (and hence framework).
- *Announcements:* it includes announcements to be published by the institution to all of its staff members, instructors, teaching assistants, students and all other related members.
- *Discussion groups:* This provides an area shared in the Internet institution site for all users to discuss any idea or ask question the other members might answer.
- *Electronic Library:* This feature might be considered as a secondary or optional feature to the framework. It is not a core requirement. (It is only considered as an option).
- *Bulletin Boards:* It is considered as an administrative service. Used to communicate announcements between the administration, staff members, academic members (instructors, teaching assistants), and students to each other.
- *Electronic Agenda:* Three types of agendas are to be considered in this feature they are: institutional agenda, departmental agenda, and course agenda. Where the institutional agenda is a set of deadline dates for the institution as a whole (a calendar), the departmental agenda is a calendar of academic meetings, calendar, conferences, and other actions on a semester basis for a specific department, and finally the course agenda is intended to provide its related users with a deadline dates for exams, projects, presentations, assignments for a specific course (it may include a calendar for lectures presentations).
- *Chat-rooms services:* This feature includes an online chat service for the framework end users to communicate their ideas and any questions online with others.
- *Emailing Services:* This service should provide to its members an email address specific to the institution they belong to with all of its corresponding utilities.

- *Related Links Services:* This feature includes providing the students with all related links to a specific topic or material browsed by the student while studying (learning).
- *Assignments Turn-in Services:* This feature might be used to allow users to turn in their assignments and projects electronically to the their instructors.

## REFERENCES

1. [Pree-Pomberger-Kapsner] Pree, W., Pomberger, G., and Kapsner, F. Framework Component Systems - Concepts, Design, Heuristics, and Perspectives: pp. 1-11
2. [Fontoura-Crespo-Lucena-Alencar-Cowan1999] Fontoura, M., Crespo, S., Lucena, J.C., Alencar, P., and Cowan, D. Using viewpoints to derive object-oriented frameworks: a case study in the web-based education domain. *The journal of Systems and Software (JSS)*. December 1999: pp. 239- 257
3. [Ebner-Shao-Tsai2000] Ebner, E., Shao, W., and Tsai W. The five-Module Framework for the Internet Application Development. *ACM*, 2000: pp. 1-7
4. [Fayad-Schmidt1997] Fayad, M.E., and Schmidt, D.C. Object-Oriented Application Frameworks. *Communications of the ACM* 40(10), October 1997: pp. 32-38
5. [Fayad-Hamu2000] Fayad, M.E., and Hamu, D.S. Enterprise Frameworks: Guidelines for Selection. *ACM Computing Surveys* Vol.32, No. 1, March 2000: pp. 1-5
6. [Fayad-Hamu-Brugali2000] Fayad, M.E., Hamu, D.S., and Brugali D. Enterprise Frameworks Characteristics, Criteria, and Challenges. *Communications of the ACM* Vol. 43, No. 10, October 2000: pp. 39-46
7. [Hamu-Fayad1998] Hamu, D.S., and Fayad, M.E. Achieving Bottom-Line Improvements with Enterprise Frameworks. *Communications of the ACM* Vol. 41, No. 8, August 1998: pp. 110-113
8. [Pugh-Leung1998] Pugh, J.R., and Leung, C. Application Frameworks: Experience with MacApp, *ACM*, 1998: pp. 142-147
9. [Brown-Wallnau1996] Brown, A.W., and Wallnau, K.C. Engineering of Component-Based Systems. *IEEE*, 1996: pp. 7-15
10. [Krause1997] Krause, L. A Framework Approach. *OOPSLA Conference on Developing Successful Object-Oriented Frameworks*, 1997: pp. 56-65
11. [Fuentes-Troya1999] Fuentes, L., and Troya, J.M. A Java Framework for Web-Based Multimedia and Collaborative Applications. *IEEE Internet Computing*, March-April 1999: pp. 55-64
12. [Artim-Leung1997] Artim, J., and Leung, T. The Definition of Framework: A usage Survey and Proposal. *OOPSLA Conference on Developing Successful Object-Oriented Frameworks*, 1997: pp. 5-8
13. [Cantu] Cantu, M., Internet Programming with Delphi: *Borland Inprise*, [www.marcocantu.com](http://www.marcocantu.com)
14. [Sommwerville1996] Sommwerville, I. *Software Engineering* – Fifth Edition, Addison-Wesley. 1996: pp. 395-417, 545-563 (Chapter 20 Software Reuse, Chapter 27 Software Engineering Environments)
15. [Alencar-Cowan-Crespo-Fontoura-Lucena1998] Alencar, P., Cowan, D., Crespo, S., Fontoura, M., and Lucena, C. Pragmatic Issues Behind Framework Design, *workshop#12 – Pragmatic Issues in Using Frameworks Implications for Framework Design – Object-oriented Programming, Systems, Languages, and Applications (OOPSLA' 98)*, Vancouver, Canada, 1998: Pp. 1-5. [www.almaden.ibm.com/cs/people/fontoura/resume.html](http://www.almaden.ibm.com/cs/people/fontoura/resume.html)
16. [Fontoura-Heausler-Lucena1998] Fontoura, M., Heausler, E., and Lucena, C. A Framework Design And Instantiation Method Based on Viewpoints. *Object-oriented Programming, Systems, Languages, and Applications (OOPSLA' 98)*,

Vancouver, Canada, October 1998: pp. 1-37.  
[www.les.inf.puc.br/~mafe/oopsia.htm](http://www.les.inf.puc.br/~mafe/oopsia.htm)

17. [Fontoura1998] Fontoura, M. A Framework Design and Instantiation Method, 1998: pp. 1-4. (Doctoral Symposium)
18. [Crespo-Fontoura-Lucena1998] Crespo, S., Fontoura, M., and Lucena, C. AulaNet: An Object-Oriented Environment for Web-Based Education. *International Conference of the Learning Sciences (ICLS'98)*, 304-306, Atlanta, USA. 1998: pp.1-3
19. [Lucena-Fuks-Milidiu'-Macedo-Santos-Laufer-Ribeiro-Fontoura-Noya-Crespo-Torres-Daflon-Lukowiecki1998] Lucena, C., Fuks, H., Milidiu', R., Macedo, L., Santos, N., Laufer, C., Ribeiro, M., Fontoura, M., Noya, R., Crespo, S., Torres, V., Daflon, L., and Lukowiecki, L. AulaNet: An Environment for the Development and Maintenance of Courses on the Web. *International Conference on Engineering Education (ICEE' 98)*, Rio de Janeiro, Brazil, 1998: Pp. 1-10.  
[www.almaden.ibm.com/cs/people/fontoura/resume.html](http://www.almaden.ibm.com/cs/people/fontoura/resume.html)
20. [Vaupe-Sommer1997] Vaupel, J., and Sommer, M. Multimedia Education, Distance Learning and Electronic Commerce Applications. *International Conference on Virtual Systems and Multimedia (VSMM '97) IEEE*, 1997: pp. 1-2
21. [Alencar-Cowan-Crespo-Fontoura-Lucena1998] Alencar, P., Cowan, D., Crespo, S., Fontoura, M., and Lucena, C. OwlNet: An Object-Oriented Environment for WBE. *Second Argentine Symposium in Object-Oriented (ASOO' 98)*, SADIO, 91-100, Buenos Aires, Argentina, 1998.  
[www.almaden.ibm.com/cs/people/fontoura/resume.html](http://www.almaden.ibm.com/cs/people/fontoura/resume.html)
22. [Pree2000] Pree, W. Object-Oriented Design Patterns and Host Spot Cards: *Applied Computer Science, University of Constance, Germany*. Pp.1-8
23. [Fontoura-Pree-Rumpe2000] Fontoura, M., Pree, W., and Rumpe, B. UML-F: A Modeling Language for Object-Oriented Frameworks. *European Conference on Object Oriented Programming (ECOOP 2000)*, Lecture Notes in Computer Science 1850, Springer, 63-82, Cannes, France, 2000: Pp. 1-22.  
[www.almaden.ibm.com/cs/people/fontoura/resume.html](http://www.almaden.ibm.com/cs/people/fontoura/resume.html)
24. [Fontoura-Moura-Crespo-Lucena1998] Fontoura, M.F., Moura, L., Crespo, S., and Lucena, C.J. ALADIN: An Architecture for Learning-ware Applications Design and Instantiation, October 1998: pp. 1-18
25. [Johnson-Foote 1991] Johnson, P., and Foote, B. Designing Reusable Classes, *Journal of Object-Oriented Programming*, Aug 1991: pp. 1-27
26. [Johnson] Johnson, R. Documenting Frameworks using Patterns, *University of Illinois at Urbana-Champaign, Department of Computer Science*: pp. 1-14
27. [Fontoura] Fontoura, M.F. Object-Oriented Applications Frameworks: The Untold Story, *Software Engineering Laboratory (LES), Computer Science Department, Pontifical Catholic University of Rio de Janeiro*: pp. 1-5
28. [Alencar-Cowan-Nelson-Fontura-Lucena] Alencar, P., Cowan, D., Nelson, T., Fontoura, M.F., and Lucena, C. Viewpoints and Frameworks in Component-Based Software Design, *Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, and Department of Computer Science, Pontifical Catholic University of Rio de Janeiro*: pp. 1-3
29. [Crespo-Fontoura-Lucena1998] Crespo, S., Fontoura, M.F., and Lucena, C. A framework Development method using viewpoints and the views-a relationship in the OO design, *Rio de Janeiro, Brazil*: pp.1-10

30. [Pree1997] Pree, W. Component-Based Software Development - A new Paradigm in Software Engineering: *Software-Concepts and Tools*, 1997.
31. [Pree]Pree, W. Essentials of Component-Ware, *Applied Computer Science, University of Constance, GGermany*: pp. 1-3
32. [Fontoura-Pree-Rumpe2000] Fontoura, M., Pree, W., and Rumpe, B. The UML Profile for Framework Architectures, *ECOOP, Cannes, June 2000*: pp. 1-43 (Slides)
33. [Crespo-Fontoura-Lucena] Crespo, S., Fontoura, M.F., and Lucena, C. Using Viewpoints, Frameworks, and Domain-Specific Languages to Enhance Software Reuse, *Rio de Janeiro, Brazil*: pp. 1-4
34. [Alencar-Cowan-Crespo-Fontoura-Lucena1998] Alencar, P., Cowan, D., Crespo, S., Fontoura, M.F., and Lucena, C. Using Viewpoints to Derive a Conceptual Model for WBE Environment, *Computer Science Group, University of Waterloo, Waterloo, Ontario, Canada, April 1998*: pp. 1-24
35. [Hancock] Hancock, J. Application Frameworks Before System Frameworks, *Patternware, Inc. NewYork, OOPSLA, 2000*: pp. 1-2
36. [Kwon-Kim-Jun-Kim2000] Kwon, I., Kim, C., Jun, J., and Kim, S. Building Generic Data Interface Components through a Data Object Generalization Patter, *Journal of Object-Oriented Programming (JOOP)*, Vol. 13, No. 6, October 2000: pp. 6-10
37. [Pree] Pree, W. Hot-Spot-Driven Framework Development, *Software Research Lab, University of Constance, Germany*: pp. 1-13
38. [Pang2000] Pang, C. A Framework Pattern for Transaction Handling and Dynamic Object Binding in a Component Broker-Based Architecture, *Journal of Object-Oriented Programming (JOOP)*, Vol. 13, No. 6, October 2000: pp. 26-31
39. [Liao-Cheung-Liu1999] Liao, S., Cheung, L., and Liu, W. An Object-Oriented System for the Reuse of Software Design Items, *Journal of Object-Oriented Programming (JOOP)*, Vol. 11, No. 8, January 1999: pp. 22-28
40. [Fontoura1999] Fontoura, M.F. A systematic Approach for Framework Development (Thesis), *Dissertation presented to the Computer Science, Computer Science Department, Pontifical Catholic University of Rio de Janeiro*, July 1999: pp. 1-165
41. [Fayad-Schmidt-Johnson1999] Fayad, M.E., Schmidt, D., and Johnson, R. *Building Application Frameworks: Object-Oriented Foundations of Framework Design*. John Wiley & Sons, New York, September 1999: pp. 3-86, 153-160, 620-624 (Chp1, 3 & Side-Bar 2, Side-Bar 9)
42. [Fayad-Johnson1999] Fayad, M.E., and Johnson, R. *Domain-Specific Application Frameworks: Frameworks Experience by Industry*. John Wiley & Sons, New York, October 1999: pp. 437-469, 615-632 (Chp.21: MultiTel, Chp. 29: Framework: A survey)
43. [Elmasri-Navathe2000] Elmasri, R., and Navathe, S. *Fundamentals of Database Systems (Third Edition)*, Addison Wesley Series, 2000
44. [Fedorov-Elmonona2000] Fedorov, A., and Elmonona, N. *Advanced Delphi Developer's Guide to ADO*, Word-ware Publishing Inc., 2000
45. *TCP/IP*, Microsoft Press, 1998
46. [www.cs.odu.edu/~wild/docs/spiral.html](http://www.cs.odu.edu/~wild/docs/spiral.html)
47. [Pressman 2001] Pressman, R. *Software Engineering: A practitioner's approach*. Mc Graw Hill, New York, 2001. (Fifth Edition). ISBN: 0-07-365578-3.

48. [Booch–Rumbaugh–Jacobson1999] Booch, G., Rumbaugh, J., and Jacobson, I. *The Unified Modeling Language User Guide: The ultimate tutorial to the UML from the original designers*. Addison Wesley, 1999: pp. 105-115, 233-240, 243-255, 331-338, 369-379. (Seventh printing, August 2000). (Chapters: 8, 17, 18, 24, 27) ISBN: 0-201-57168-4.
49. *Microsoft SQL Server 7.0 Database Implementation – Training Kit: Hands-On, Self-Paced Training*, Microsoft Press, 1999. ISBN: 1-57231-826-0.
50. Borland Inprise Developer's Guide Delphi 5 (for Windows 98, Windows 95, and Windows NT): *Developer's Guide*
51. [Gamma-Johnson-Vlissides1995] Gamma, E., R. Helm, R. Johnson, and Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley, 1995.



## APPENDICES

### Appendix-A – The Complete database generation script for AFDL (Data Dictionary) as implemented by Microsoft SQL Server DBMS.

```
CREATE TABLE [dbo].[AccessLevel] (  
    [ID] [int] IDENTITY (1, 1) NOT NULL ,  
    [UserGroup] [int] NOT NULL ,  
    [Table] [varchar] (50) NULL ,  
    [TableRead] [bit] NULL ,  
    [TableWrite] [bit] NULL ,  
    [StoredProcedure] [varchar] (50) NULL  
    ) ON [PRIMARY]  
GO
```

```
CREATE TABLE [dbo].[Advisor] (  
    [ID] [int] IDENTITY (1, 1) NOT NULL ,  
    [Grad_Student] [int] NOT NULL  
    ) ON [PRIMARY]  
GO
```

```
CREATE TABLE [dbo].[Area] (  
    [ID] [int] IDENTITY (1, 1) NOT NULL ,  
    [Description] [varchar] (30) NOT NULL ,  
    [Country] [smallint] NOT NULL ,  
    [State] [int] NOT NULL  
    ) ON [PRIMARY]
```

GO

```
CREATE TABLE [dbo].[City] (  
    [ID] [smallint] IDENTITY (1, 1) NOT NULL ,  
    [Description] [varchar] (30) NOT NULL ,  
    [Country] [smallint] NOT NULL  
) ON [PRIMARY]
```

GO

```
CREATE TABLE [dbo].[Class] (  
    [ID] [tinyint] IDENTITY (1, 1) NOT NULL ,  
    [Description] [varchar] (20) NOT NULL  
) ON [PRIMARY]
```

GO

```
CREATE TABLE [dbo].[Committee] (  
    [Id] [int] IDENTITY (1, 1) NOT NULL ,  
    [Faculty] [int] NOT NULL ,  
    [Grad_Student] [int] NOT NULL  
) ON [PRIMARY]
```

GO

```
CREATE TABLE [dbo].[Country] (  
    [ID] [smallint] IDENTITY (1, 1) NOT NULL ,  
    [Description] [varchar] (30) NOT NULL
```

```
) ON [PRIMARY]
```

```
GO
```

```
CREATE TABLE [dbo].[Course] (
```

```
    [ID] [smallint] IDENTITY (1, 1) NOT NULL ,
```

```
    [CNumber] [varchar] (3) NOT NULL ,
```

```
    [Interface] [tinyint] NOT NULL ,
```

```
    [Code] [varchar] (3) NOT NULL ,
```

```
    [CName] [varchar] (50) NOT NULL ,
```

```
    [Department] [int] NOT NULL ,
```

```
    [CDesc] [text] NULL ,
```

```
    [Credits] [tinyint] NOT NULL ,
```

```
    [Prerequisites] [varchar] (50) NULL ,
```

```
    [Syllabus] [text] NULL
```

```
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
```

```
GO
```

```
CREATE TABLE [dbo].[Degrees] (
```

```
    [ID] [tinyint] IDENTITY (1, 1) NOT NULL ,
```

```
    [Grad_Student] [int] NOT NULL ,
```

```
    [CollegeName] [varchar] (100) NOT NULL ,
```

```
    [Title] [varchar] (50) NOT NULL ,
```

```
    [Description] [varchar] (255) NULL ,
```

```
    [Year] [tinyint] NOT NULL
```

```
) ON [PRIMARY]
```

GO

```
CREATE TABLE [dbo].[Department] (  
    [ID] [int] IDENTITY (1, 1) NOT NULL ,  
    [DName] [varchar] (50) NOT NULL ,  
    [Institution] [smallint] NOT NULL ,  
    [DPhone] [varchar] (50) NULL ,  
    [Office] [varchar] (50) NULL  
    ) ON [PRIMARY]
```

GO

```
CREATE TABLE [dbo].[Faculty] (  
    [ID] [int] IDENTITY (1, 1) NOT NULL ,  
    [Person] [int] NOT NULL ,  
    [FPhone] [varchar] (20) NULL ,  
    [FOffice] [varchar] (20) NULL ,  
    [Femail] [varchar] (50) NULL ,  
    [Dean] [bit] NOT NULL ,  
    [Advisor] [bit] NOT NULL ,  
    [CV] [text] NULL ,  
    [OfficeHrs] [varchar] (50) NULL ,  
    [Rank] [tinyint] NULL ,  
    [Salary] [money] NULL ,  
    [Chairs] [bit] NOT NULL ,  
    [WorkStartDate] [smalldatetime] NULL ,
```

```
        [WorkEndDate] [smalldatetime] NULL ,
        [Note] [text] NULL
    ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

```
CREATE TABLE [dbo].[FacultyMember] (
    [ID] [int] IDENTITY (1, 1) NOT NULL ,
    [Faculty] [int] NOT NULL ,
    [Department] [int] NOT NULL
) ON [PRIMARY]
GO
```

```
CREATE TABLE [dbo].[Grad_Student] (
    [ID] [int] IDENTITY (1, 1) NOT NULL ,
    [College] [smallint] NULL ,
    [Degree] [tinyint] NOT NULL ,
    [Year] [smallint] NULL ,
    [Advisor] [int] NOT NULL
) ON [PRIMARY]
GO
```

```
CREATE TABLE [dbo].[Grant] (
    [ID] [int] IDENTITY (1, 1) NOT NULL ,
    [Title] [varchar] (20) NOT NULL ,
    [Faculty] [int] NOT NULL ,
```

```
[No] [int] NOT NULL ,  
[Agency] [varchar] (20) NOT NULL ,  
[StartDate] [smalldatetime] NOT NULL  
) ON [PRIMARY]  
GO
```

```
CREATE TABLE [dbo].[IDType] (  
    [ID] [tinyint] IDENTITY (1, 1) NOT NULL ,  
    [Description] [varchar] (50) NOT NULL  
) ON [PRIMARY]  
GO
```

```
CREATE TABLE [dbo].[Institution] (  
    [ID] [smallint] IDENTITY (1, 1) NOT NULL ,  
    [COffice] [varchar] (50) NOT NULL ,  
    [CName] [varchar] (50) NOT NULL ,  
    [Dean] [varchar] (50) NOT NULL ,  
    [Address] [varchar] (50) NULL ,  
    [POBox] [varchar] (10) NULL ,  
    [Phones] [varchar] (20) NOT NULL ,  
    [Fax] [varchar] (20) NULL ,  
    [Email] [varchar] (50) NULL ,  
    [Site] [varchar] (50) NULL ,  
    [Note] [text] NULL  
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
```

GO

```
CREATE TABLE [dbo].[Instructor_Researcher] (  
    [ID] [int] IDENTITY (1, 1) NOT NULL ,  
    [GradStudent] [bit] NOT NULL ,  
    [Faculty] [bit] NOT NULL ,  
    [Note] [text] NULL ,  
    [Image] [image] NULL  
    ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]  
GO
```

```
CREATE TABLE [dbo].[Interface] (  
    [ID] [tinyint] IDENTITY (1, 1) NOT NULL ,  
    [Language] [varchar] (20) NOT NULL ,  
    [SiteType] [tinyint] NOT NULL  
    ) ON [PRIMARY]  
GO
```

```
CREATE TABLE [dbo].[Major] (  
    [ID] [int] IDENTITY (1, 1) NOT NULL ,  
    [Department] [int] NOT NULL ,  
    [Description] [varchar] (50) NOT NULL ,  
    [Degree] [varchar] (35) NOT NULL ,  
    [DegreeSymbol] [char] (3) NOT NULL  
    ) ON [PRIMARY]
```

GO

```
CREATE TABLE [dbo].[Material] (  
    [ID] [smallint] NOT NULL ,  
    [TopicID] [int] NOT NULL ,  
    [Type] [tinyint] NOT NULL ,  
    [Interface] [tinyint] NOT NULL ,  
    [Content] [binary] (10) NULL ,  
    [Text] [char] (10) NULL  
    ) ON [PRIMARY]
```

GO

```
CREATE TABLE [dbo].[MaterialType] (  
    [ID] [tinyint] IDENTITY (1, 1) NOT NULL ,  
    [Code] [varchar] (10) NULL ,  
    [Description] [varchar] (30) NOT NULL ,  
    [Extension] [char] (3) NULL  
    ) ON [PRIMARY]
```

GO

```
CREATE TABLE [dbo].[Minor] (  
    [ID] [smallint] IDENTITY (1, 1) NOT NULL ,  
    [Department] [int] NOT NULL ,  
    [Description] [varchar] (20) NOT NULL  
    ) ON [PRIMARY]
```



GO

```
CREATE TABLE [dbo].[Person] (  
    [ID] [int] IDENTITY (1, 1) NOT NULL ,  
    [FirstName] [varchar] (20) NOT NULL ,  
    [MiddleName] [varchar] (20) NULL ,  
    [LastName] [varchar] (20) NOT NULL ,  
    [MotherName] [varchar] (20) NULL ,  
    [IDType] [tinyint] NOT NULL ,  
    [IDNb] [varchar] (30) NULL ,  
    [Faculty] [bit] NOT NULL ,  
    [Student] [bit] NOT NULL ,  
    [DateOfBirth] [smalldatetime] NULL ,  
    [Gender] [tinyint] NOT NULL ,  
    [AddressNo] [varchar] (50) NULL ,  
    [Street] [char] (10) NULL ,  
    [AptNo] [char] (10) NULL ,  
    [City] [smallint] NULL ,  
    [State] [int] NULL ,  
    [Country] [smallint] NULL ,  
    [Area] [int] NULL ,  
    [Zip] [char] (30) NULL ,  
    [POBox] [varchar] (10) NULL ,  
    [PostalCode] [varchar] (10) NULL ,  
    [Address] [varchar] (50) NULL ,
```

```

[HomePhone] [varchar] (20) NULL ,
[BusinessPhone] [varchar] (20) NULL ,
[MobilePhone] [varchar] (20) NULL ,
[EMail] [varchar] (50) NULL ,
[Note] [text] NULL ,
[Image] [image] NULL ,
[BloodGroup] [char] (2) NULL ,
[RH] [char] (1) NULL ,
[SOSName] [varchar] (30) NULL ,
[SOSHomePhone] [varchar] (20) NULL ,
[SOSMobilePhone] [varchar] (20) NULL ,
[MaritalStatus] [tinyint] NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

```

```

CREATE TABLE [dbo].[Rank] (
    [ID] [tinyint] IDENTITY (1, 1) NOT NULL ,
    [Description] [varchar] (20) NOT NULL
) ON [PRIMARY]
GO

```

```

CREATE TABLE [dbo].[Section] (
    [ID] [int] IDENTITY (1, 1) NOT NULL ,
    [SecNumber] [varchar] (50) NOT NULL ,
    [Course] [smallint] NOT NULL ,

```

```
[Year] [tinyint] NOT NULL ,  
[Semester] [tinyint] NOT NULL ,  
[CurrentSection] [bit] NOT NULL ,  
[Student] [int] NOT NULL ,  
[Instructor] [int] NOT NULL ,  
[EnrolledStudents] [smallint] NULL ,  
[AvailableChairs] [smallint] NULL  
) ON [PRIMARY]  
GO
```

```
CREATE TABLE [dbo].[Semester] (  
    [ID] [tinyint] IDENTITY (1, 1) NOT NULL ,  
    [Description] [char] (20) NOT NULL  
) ON [PRIMARY]  
GO
```

```
CREATE TABLE [dbo].[Services] (  
    [ID] [int] IDENTITY (1, 1) NOT NULL ,  
    [Description] [varchar] (20) NOT NULL ,  
    [Type] [tinyint] NOT NULL ,  
    [Interface] [tinyint] NOT NULL ,  
    [Course] [smallint] NOT NULL  
) ON [PRIMARY]  
GO
```

```
CREATE TABLE [dbo].[ServiceType] (  
    [ID] [tinyint] IDENTITY (1, 1) NOT NULL ,  
    [Description] [varchar] (50) NOT NULL  
) ON [PRIMARY]  
GO
```

```
CREATE TABLE [dbo].[Site] (  
    [ID] [int] IDENTITY (1, 1) NOT NULL ,  
    [Type] [tinyint] NOT NULL ,  
    [Address] [varchar] (255) NOT NULL  
) ON [PRIMARY]  
GO
```

```
CREATE TABLE [dbo].[SiteType] (  
    [ID] [tinyint] IDENTITY (1, 1) NOT NULL ,  
    [Description] [varchar] (50) NOT NULL  
) ON [PRIMARY]  
GO
```

```
CREATE TABLE [dbo].[State] (  
    [ID] [int] IDENTITY (1, 1) NOT NULL ,  
    [Description] [varchar] (50) NOT NULL ,  
    [Code] [char] (5) NULL ,  
    [Country] [smallint] NOT NULL  
) ON [PRIMARY]
```

GO

```
CREATE TABLE [dbo].[Student] (  
    [ID] [int] IDENTITY (1, 1) NOT NULL ,  
    [Class] [tinyint] NOT NULL ,  
    [Section] [int] NULL ,  
    [Major] [int] NOT NULL ,  
    [Minor] [smallint] NULL  
    ) ON [PRIMARY]
```

GO

```
CREATE TABLE [dbo].[Support] (  
    [ID] [int] IDENTITY (1, 1) NOT NULL ,  
    [Faculty] [smallint] NOT NULL ,  
    [Instructor] [int] NOT NULL ,  
    [StartDate] [smalldatetime] NOT NULL ,  
    [Time] [smalldatetime] NOT NULL ,  
    [EndDate] [smalldatetime] NOT NULL  
    ) ON [PRIMARY]
```

GO

```
CREATE TABLE [dbo].[Topic] (  
    [ID] [int] IDENTITY (1, 1) NOT NULL ,  
    [CourseID] [smallint] NOT NULL ,  
    [ParentID] [smallint] NOT NULL ,
```

```
        [Interface] [tinyint] NOT NULL ,
        [Title] [varchar] (100) NOT NULL ,
        [Description] [text] NULL
    ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

```
CREATE TABLE [dbo].[Transcript] (
    [ID] [int] IDENTITY (1, 1) NOT NULL ,
    [Student] [int] NOT NULL ,
    [Section] [int] NOT NULL ,
    [Grade] [char] (1) NULL ,
    [Note] [text] NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

```
CREATE TABLE [dbo].[User] (
    [ID] [int] IDENTITY (1, 1) NOT NULL ,
    [UserName] [varchar] (50) NOT NULL ,
    [Password] [varchar] (12) NULL ,
    [UserGroup] [int] NOT NULL
) ON [PRIMARY]
GO
```

```
CREATE TABLE [dbo].[UserGroup] (
    [ID] [int] IDENTITY (1, 1) NOT NULL ,
```

```
[Description] [varchar] (50) NULL  
) ON [PRIMARY]  
GO
```

**Remark:** A reference to the data dictionary is also provided as a soft copy using the MS-SQL Server version 7.0 – It could be provided upon request at the email address: [ahmadnatafgi@hotmail.com](mailto:ahmadnatafgi@hotmail.com)