UML and Modeling Sites for the World Wide Web

J. Drew Procaccino[*], Marvin E. Darter[*], Il-Yeol Song[**], and Gloria Wai-Min Tang[**]

*Rider University
College of Business Administration
2183 Lawrenceville Road
Lawrenceville, New Jersey 08648 USA
E-mail: jdproc@aol.com

**Drexel University
College of Information Science & Technology
3141 Chestnut Street
Philadelphia, Pa. 19104-2875 USA

Abstract

Using a case study approach, this paper introduces and outlines the *Unified Modeling Language* (UML) as it applies to modeling a site on the World Wide Web. The authors include an introduction to the concept of modeling, in general, as well as how modeling relates to the design of a Web site. A simple, fictitious university Web site serves as an illustrative tool throughout the paper. This site is reflected in several UML-based diagrams, as well as the discussion of some of the issues, considerations and techniques when using UML to model a Web site. The paper concludes with a list of 'best practices' when modeling Web sites using UML.

1. Introduction

This paper introduces and outlines the *Unified Modeling Language* (UML) as it applies to modeling a site on the World Wide Web. We focus our attention on Internet-based systems, essentially all of the UML-related design implications included in our discussion hold for Intranet-based systems, as well. UML is more than able to model complex, Web-based applications, including transaction processing (such as book/CD ordering system) and document management (such as an academic conference manager). However, for purposes of simplicity, we will be discussing some of the major considerations of modeling a relatively simple Web site using UML, largely from the user's perspective (client-side). Our examples will be drawn from the Web site of a fictitious University (see Appendix A), and we include several UML-based diagrams, which are intended to illustrate various points of modeling with UML. We used *Rational Software Corporation's*, Rational Rose, to generate our UML diagrams, and combined with a narrative of the site, help to illustrate the concepts being proposed and

the site being developed.

Booch, Rumbaugh and Jacobson [1999] define UML as a "standard language for writing software blueprints", including the capability to "visualize, specify, construct and document the artifacts" of the system to be modeled through the use of numerous diagrams [Booch, et al, 1999]. UML offers consistent notations and numerous tools across processes and projects. Jim Conallen, Web Modeling Evangelist at Rational Software Corporation, suggests that UML is the "language of choice for modeling software-intensive systems" [Conallen, 1999]. Web site development falls into this category of 'software-intensive systems'.

In illuminating the specific merits of UML when modeling Web sites and applications, Conallen pointed out the following [Conallen, 1999]:

- Web applications are a type of software-intensive system that are not only becoming increasingly complex, but are being implemented in more critical situations.
- A software system typically has multiple models, each representing a different viewpoint, level of abstraction and detail.
- The proper level of abstraction and detail depend on the artifacts and worker activities in the development process.
- UML can "express the execution of the system's business logic [if any] in those Web-specific elements and technologies" [Conallen, 1999].

UML can model specific representations at various levels of abstraction. These different levels are comprised of several diagrams, which taken together, allow us to view the design of the system with as much, or as little, detail as needed. Modeling at differing levels of abstraction (between high-level/generalized and low-level/detailed) will depend on exactly what information needs to be conveyed through the completed model.

When modeling any system, Conallen [1999] suggests the importance of concentrating on the elements that will be of value to those who will be using the model (system developers). This entails "model [ing] the artifacts of the system – those 'real life' entities that will be constructed and manipulated to produce the final product". Of course, the artifacts of a particular system will depend on the system being modeled. Artifacts of a Web site may include, but are not limited to, the following:

- Web pages,
- Multimedia-based elements, such as images, animation, video and audio clips,
- Inter- and intra-page hyperlinks ("navigational paths" through a Web site),
- Dynamic Web page content, both on the client and server-side,
- Various end-users of the system.

Depending on the particular site, these elements, or a subset of them, are of direct concern to the designers and creators of a Web site. In general, modeling the internal workings of the Web server or Web browser will not lend any significant insight to the designers and creators (programmers) of the site, and as such, would not be included in a typical UML diagram. Given the characteristics of our sample University Web site, we felt that modeling the navigational links and paths of the site was a priority. Among software-based systems, a site map is unique to a Web-based system and UML's corresponding tool for this map is a Component Diagram, which is discussed later in this paper.

The structure of this paper is as follows. We begin with a general introduction to

modeling, including why we model and what UML represents. This section also includes a general architecture of a Web site, as well as an overview of a fictitious sample University Web site. We then present more traditional approaches to modeling a Web site, including the development of a cognitive walkthrough and a storyboard. Our next major section presents some of the issues, considerations and techniques when using UML to model a Web site. This section makes extensive use of various UML-based diagrams. Lastly, we present our conclusions, which include our list of 'best practices' when modeling Web sites using UML.

1.3 Scope of our analysis: simplified university Web site

Regardless of the modeling method(s) and tools employed, there are several critical aspects to designing an effective, easy-to-navigate and information Web site. It is critical for a Web site to provide information and support the functions that its users need. These aspects are related to who is expected to use the site and what tasks these specific users need to accomplish through the site. Specifically, an initial analysis needs to be completed of at least the following:

- Determine the overall purpose of the site.
- Identify the intended users of the site.
- Frame the scope of information contained within the site.

The overall purpose of our sample University Web site is twofold. The first is to provide information related to programs, people and admissions. The second is to facilitate contact through phone, mail and e-mail between users of the site, and representatives of the University (faculty and administration). Brannan [2000] suggested that part of the design process to is attempt to identify groups of users based on their common informational needs, and then essentially generate a navigation and manipulation model from the information gathered. This would result in applications that are more tailored to users. The intended users of the site are as follows:

- Potential students and their parents/guardians.
- Current students and their parents/guardians.
- Faculty and administrators.
- Industry representatives.
- Alumni.

There may be inherent variety among these users in terms of their expectations, goals and technical constraints (including the speed of their modem and Internet hookup) [Baresi, 2000].

The information contained in the site pertains to the following:

- Academic programs (undergraduate, graduate and continuing education).
- People associated with the University (faculty, administration and alumni).
- Admission information (undergraduate, graduate and continuing education).

1.4 Traditional methods of Web site modeling

There are a few traditional methods for modeling a Web site. They include the following:

- Text-based description of the general contents and navigational requirements of the site.
- Cognitive walkthroughs for each user task.
- Storyboard of the site.

The following section includes text-based description of the general contents and navigational requirements of the site. It should be noted that for brevity, this description includes only the home page, the major sections of the site and subsequent 'first-level' sub-pages. This site contains approximately 50 pages, which, with the exception of the home page, are arranged into the following categories: **Programs**, **People** and **Admissions**. Appropriate links on each of their pages are shown as follows: PageName. The Programs section of the site includes pages for Undergraduate (Overview, Majors/Minors, Class List), Graduate (Overview, Majors/Minors, Class List) and Continuing Education (Overview, Class List). The People section of the site includes pages for Faculty (Overview, Faculty List), Administration (Overview, Administrator List) and Alumni (Overview, Alumni List). The Admissions section of the site includes pages for Undergraduate (Overview, Apply), Graduate (Overview, Apply) and Continuing Education (Overview, Apply). Each of these pages links to their respective Overview and Apply pages. Every page in the site contains a link back to the Home Page, as well as a link page to the appropriate section of the site (Programs, People or Admissions).

Cognitive walkthrough provides step-by-step instructions, combined with prototyped screens, to test the completeness of the site in executing a given user task. Similar walkthroughs would need to be developed and documented for other common user tasks. The specific tasks would need to be determined through user studies, perhaps in the form of interviews and/or surveys. A storyboard is used to illustrate the navigational hierarchy and paths within a Web site. The direction of the various arrows indicate the destination page of a particularly hyperlink.

The following sections relate to UML-specific modeling, with some general discussions that are supplemented by diagrams and documentation that is specific to our University Web site.

2. Implementing UML in Web site modeling

We have selected those diagrams that we deemed to be most relevant to modeling a Web site, particularly those with extensive navigation path analysis. We include discussions of the following elements and diagrams of UML, arranged by the following general and specific categories:

- ***System Analysis:***
  - 2.1 Problem Statement
  - 2.2 Use Case Diagrams
  - 2.3 Analysis-Level (high-level) Class Diagrams
- ***System Design:***
  - 3.1 Sequence Diagrams
  - 3.2 State Diagrams
  - 3.3 Activity Diagrams
  - 3.4 Design-Level (low-level) Class Diagrams
- ***Physical Design:***
  - 4.1 Component Diagrams
  - 4.2 Deployment Diagrams
- ***Applications Design:***
  - 5.1 Interaction Diagrams

This section will investigate and demonstrate how UML is used to model the design of a Web site, with appropriate levels of abstraction. We will be developing a primary Use Case to serve as a basis when utilizing UML and producing our UML-based examples. We will not attempt to model, in detail, these 'backend' aspects of the Web's client/server architecture. For simplicity, we will also not attempt to model any animation or real time graphic techniques, such as 'mouse-over' help.)

2. SYSTEM ANALYSIS

2.1 Problem Statement

Our assignment is to develop a Web site for a University that will provide pertinent information to a wide variety of users. These users include potential students, students, parents and guardians, faculty/administrators, alumni and representatives from industry. The included information relates to information on Programs, People and Admissions. The site should also assist users in contacting various representatives of the University (including faculty, administration, admissions and alumni) through e-mail links and contact information (address, telephone and fax numbers). The site must also provide for password-restricted access by enrolled students to pages containing course grades and assignments.

2.2 Use Case Diagrams

Before discussing Use Cases, we present a discussion of the concept of mapping user groups to UML's Actor object. We have previously identified our user groups and they can be directly mapped to Actors in UML. Actors are identified based on their distinctive interactive role with the system being modeled. For the purpose of modeling, therefore, Actors are considered external to the system. The operational and navigational needs of various user groups are "associated with the actors [that] they are specific to" [Baresi, et al, 2000]. The identified Actors (users) of our sample University Web site are: prospective students, students, parents and guardians, faculty and administration, alumni, and industry representatives.

Rosenberg [1999] defines a use case as "a sequence of actions that an Actor performs within a system to achieve a particular goal". (For clarity, we would reword this to read, "…that an Actor performs through a system…" We believe that this wording would more accurately represent the role of the Actor(s)). We have identified two "analysis-level (or business process)" Use Cases related to our University Web site [Rosenberg, 1999]:

1. Access various information regarding the Programs, People and Admissions of the University. Relevant Actors for this Use Case include potential students, current students, students' parents/guardians, faculty, administrators, alumni and industry representatives.

2. Contact various representatives of the University, including faculty and administrators, primarily through e-mail. Relevant Actors for this Use Case include potential students, current students, students' parents/guardians, faculty, administrators, alumni and industry representatives.

The specific detailed, or design-level, Use Case that we will use for purposes of illustration is the following. A potential student of the University is interested in accessing an overview of the "Introduction To Computing", an undergraduate course offered through the College of Information Technology. There are several available forms of Use Case documentation. We have used a template developed by Dr. Il-Yeol Song (College of Information Science & Technology, Drexel University USA) to present formal, structured, high-level descriptions of our Use Cases. Figure 1 includes these Use Case Descriptions.

Figure 1: Use Case Descriptions

Level	Access Information Use Case	Contact Representative Use Case
Primary (1)	1a. Access Information	1b. Contact Representative (not detailed)
Secondary (2)	2a.1. Access Restricted Information 2a.2. Access General Information	2b.1. Contact Faculty (not detailed) 2b.2. Contact Administration (not detailed) 2b.3. Contact Admissions (not detailed)
Ternary (3)	3a.1.Access Overview for Introduction To Computing Course (not detailed)	3b.1a. Contact Dr. Smith (not detailed)

Use Case Reference #	1a
Use Case Name	Access Information
Actor	Currently enrolled students.
Purpose	To access grades and assignments of a particular course.
Overview and scope	Students use the various pages and navigational links to access information on the grades and assignments of the courses they are currently enrolled in.
Level	Primary
Preconditions	Connection to the World Wide Web through an HTTP connection. Access to the University's Web site.
Post conditions in words	Desired information is displayed on the student's screen.
Trigger	A student wants to access their course grade(s) or assignment(s).
Included Use Cases	None.
Extension Use Cases	Access Restricted Information. Access General Information.
Frequency	Unknown.
Other Comments	This is a high level Use Case description, as the level of detail shows. 'Primary' Level refers to a top-level Use Case. Access Information and Contact Representatives are each primary level Use Cases.

Use Case Reference #	2a.2
Use Case Name	Access General Information
Actor	Potential students, students, parents & guardians, faculty & administrators, alumni and industry reps.
Purpose	To access information regarding the University.
Overview and scope	Actors use the various pages and navigational links to access information on the University's Programs (Undergraduate, Graduate and Continuing Education), People (Faculty, Administration and Alumni) and Admissions (Undergraduate, Graduate and Continuing Education).
Level	Secondary
Preconditions	Connection to the World Wide Web through an HTTP connection. Access to the University's Web site.
Post conditions in words	Desired information is displayed on the Actor's screen.
Trigger	An actor wants to access information about the University.
Included Use Cases	None.
Extension Use Cases	None.
Frequency	Unknown.
Other Comments	This is a secondary Use Case description, as the level of detail shows. 'Secondary' Level refers to a second-level Use Case, specifically Access Information: Access General Information.

A set of Use Case Diagrams provides a high-level view of the operational and navigational aspects of the site [Baresi, et al, 2000]. Our University example is a relatively straightforward in its structure, navigation and operation. Therefore, we can model all of our access paths in one diagram. Figure 2 depicts our Use Case Diagram for our Web site. This high level of Use Case does not differentiate between client and server-side functions. It includes the following:

- Students Actors can be specialized into:
  - *Students In Introduction To Computing*: This refers to those students that are currently enrolled in the course. These students will have password-enabled access to the Grades & Assignment pages of Introduction To Computing.
  - *Students Not In Introduction To Computing*: This refers to all other students. These students will only be able to access those pages that are not password protected.

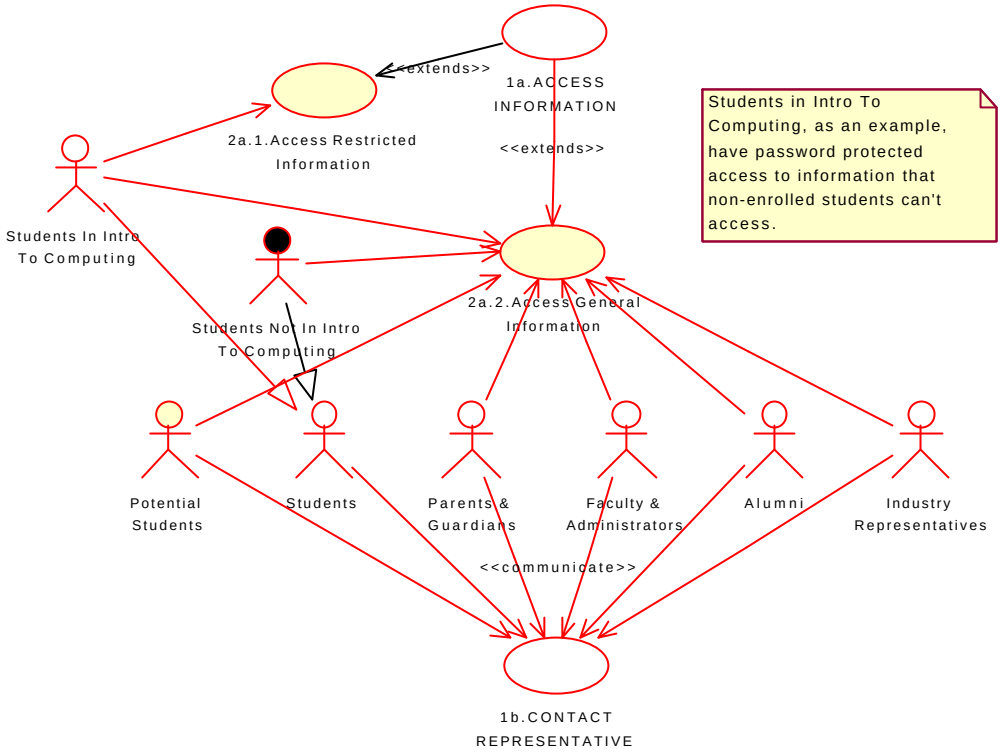
- The Access Information Use Case is extended into:
  - *Access Restricted Information Use Case*: This refers to those pages that are restricted to people with the appropriate password. An example is the Grades & Assignment page of the Introduction To Computing page.
  - *Access General Information Use Case*: This refers to those pages that can be accessed by any of the identified Actors.

Baresi and his colleagues [2000] suggested that requirements of Web sites that can be represented by Use Case Diagrams fall into two classes: operational and navigational. Operational requirements include those functions that "modify the state of the applications". Transaction processing would be an example of an operational requirement. In our University site, we consider using the site to e-mail representatives of the University as an operational requirement. Navigational requirements refer to the various interactions between the Actors and the site. Operational and navigational can both be represented through the use of Use Case Diagram(s) by one of two methods:

a. Separate models for navigational and operational requirements,
b. A single, combined model that includes color-coding of the two classes within one diagram.

The choice between the two methods depends on "the degree of intertwining between operations and navigations". Due to its relative simplicity, our University site can be modeled using one color-coded diagram. We specialized Student Actors into "Students In Intro To Computing" and "Students Not In Intro To Computing". However, the remaining Actors also do not have access to the restricted Introduction To Computing pages. Therefore, based on restricted access to the pages related to this course, we can create two distinct groups of users for this particular Use Case. Essentially, the two groups are those that have access to the restricted course pages associated with the Introduction To Computing course and those that do not. It should also be noted that we have chosen to use the <<extends>> notation to depict the Access Restricted Information and Access General Information secondary Use Cases of Access Information, the primary Use Case. We chose not to use <<includes>> because the two secondary Use Cases are not invoked by any other Use Cases.

Figure 2: 2nd Level Use Case Diagram



Grouping various Actors together based on common functionality relates to a broader UML modeling concept called 'Packaging'. This concept relates to the grouping of common objects with Packages. This tool allows the system developer to group various objects that conceptually 'fit' together. Such a grouping is intended to increase the clarity of various diagrams. In our University site and specified Use Case, the Actors can be grouped according to their access to the Introduction to Computing course's restricted Grades and Assignments pages. As mentioned, these two pages are restricted to those students currently taking the course. Other Actor-oriented packages could conceivably be developed, depending on the specified criteria of the package (a common need to access differing information or a common need to communication with differing representatives of the University, for example), as determined by the specific Use Case being analyzed. See Figure 3, which details our Actor-based Packages.

Package A: Actors who can access all pages, including those password-protected pages.	**Package B: Actors who can access all pages except the restricted Intro To Computing pages.**
Actors with access to entire site	Actors with access only to non-password protected pages
Included Actors: Students currently in Intro To Computing	*Included Actors:* All other Students Potential Students Parents & Guardians Faculty & Administrators Alumni Industry Representatives

Figure 3: Packages of Actors based on access to password-protected course pages

In fact, all of the other Actors, aside from those students currently enrolled in the Introduction To Computing course, do not have access to that course's password protected pages. Given this criterion, there is no need to detail Parents/Guardians, Faculty/Administrators and Alumni as individual Actors because all of these groups share common access (or lack thereof) to the pages related to the Introduction To Computing course. Specifically, no Actor other than those students currently enrolled in the class can access the pages related to grades and assignments.

In the absence of Packages, any differences related to operational or navigational constraints, if any, among the different Actors (users) would be noted with a comment on the navigational link to that particular Actor(s) [Baresi, et al, 2000].

2.3 Analysis-Level (high-level) Class Diagrams:

UML maps various components and entities of the project at hand to objects. Class Diagrams depict the "structures, navigations and operations" of the identified objects that users of the system utilize in order to "accomplish their tasks". Baresi and his colleagues [2000], suggested that they should be modeled from the perspective of the various users (Actors), as opposed to an implementation (physical) view. Baresi, et al, suggest that this approach might result in class diagrams that are not 'typical' of the classes that are derived from "traditional object oriented design". Further, they suggest that since these classes are modeled from the Actor's perspective, several class diagrams may be necessary in order to fully capture the context and essence of the viewpoint of the various users of the site. For purposes of illustration, we have developed a Class Diagram only of our design-level Use Case.

Baresi, et al, [2000] also suggested a need for 'navigational nodes' of a Web site to be modeled as a class. These nodes could be identified as the start and end points for users to navigate through the system. Basically, the nodes are individual Web pages, or "well identifiable logical blocks in a page" (or intra-page links) [Baresi, et al, 2000].

Figure 4 is our version of an analysis-level (high-level) Class Diagram for our University Web site. It includes both client and server collaborations.

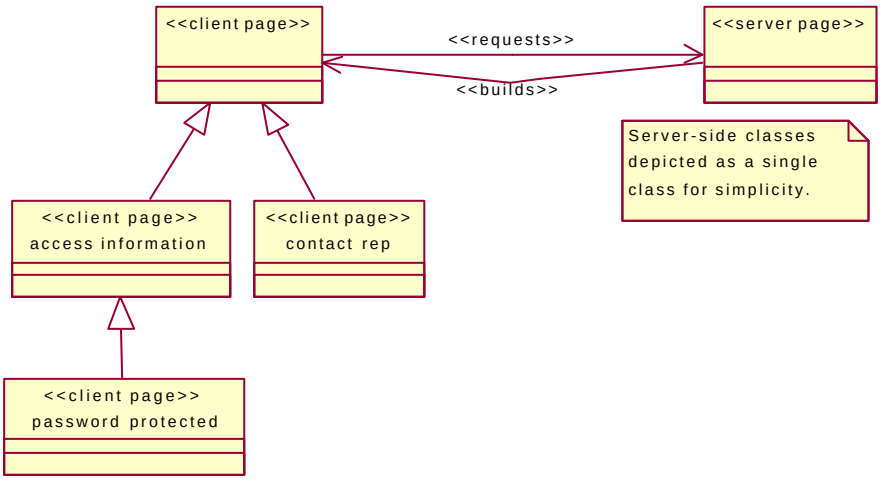


Figure 4: Analysis-Level Class Diagram

3. SYSTEM DESIGN

3.1 Sequence Diagrams

A Sequence Diagram, or a set of Sequence Diagrams, charts the steps, in order, that are necessary to complete a specific Use Case, including "all alternative courses" of action within the Use Case [Baresi, et al, 2000]. The particular Diagram being constructed determines the relative level of detail of the steps. Sequence diagrams include *boundary*, *control* and *entity* objects, as well as the narrative steps from a particular Use Case description. Our sample University site's objects can be mapped to the following:

- Boundary Object: various pages of the site (example: home page)
- Control Object: various hyperlinks (example: Programs, People and Admissions)
- Entity Object: various text of hyperlinks (example: Programs, People and Admissions)

Although not part of our state design-level Use Case (namely accessing the Overview of the Introduction To Computing Course), we wanted to include an outline of the steps necessary to gain access to one of the password-protected pages (namely, Grades). Figure 5 is a partial depiction of the Sequence Diagram that includes password-enabled access to the Grades page.

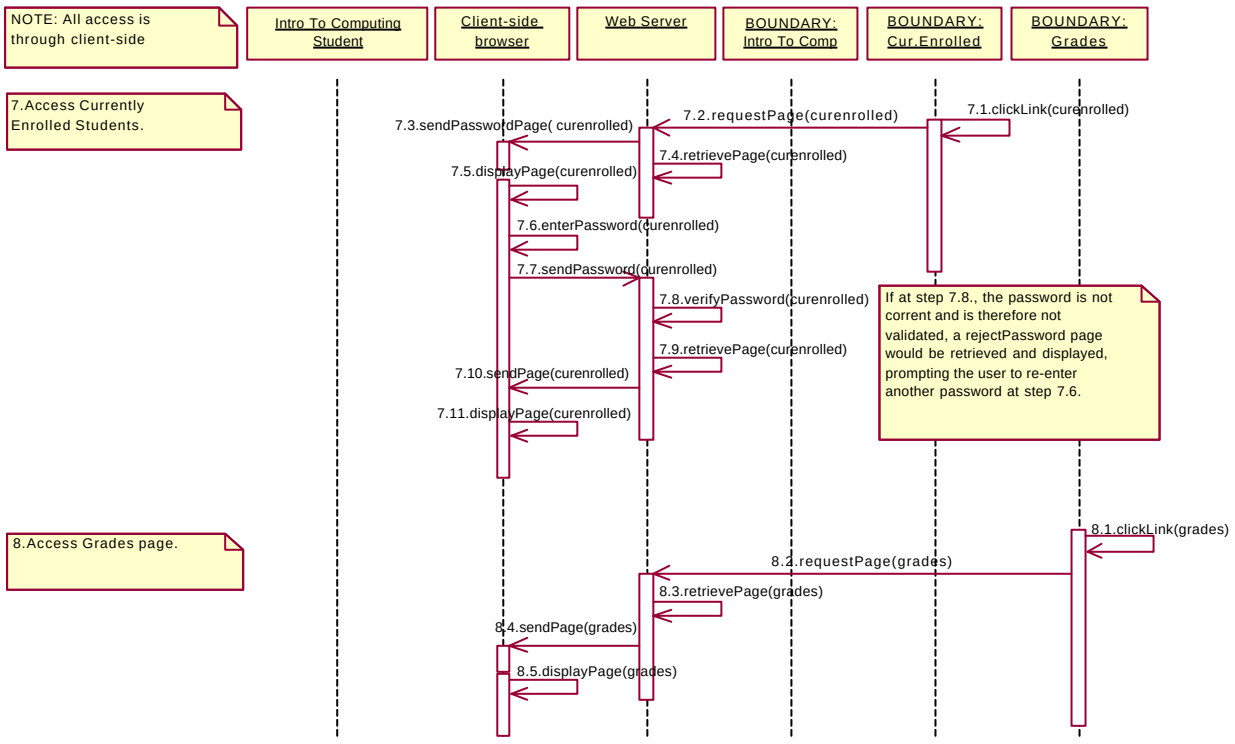


Figure 5: Partial Sequence Diagram depicting password-enabled access

Notes:

Steps 1 through 6: Access Web site home page, access Programs page, access Undergraduate page, access Class List page, access Information Technology Classes page and access Introduction To Computing page.

The various client-side browser windows are 'boundary' objects in this Use Case. The Web server object is a 'control' object. The format of the messaging resembles a 'stair' format, whereby there is a delegation of authority among the object 'lifelines'.

UML includes a conceptual view that consists of several diagrams, and it is called the Implementation View. We include the following diagrams as part of our University Web site analysis: static-based views (Component Diagrams) and dynamic-based views (State Diagram and Interface Design Diagram).

3.4 Design-Level (low-level) Class Diagrams

We have added operations and attributes to our University Web site in Figure 6, as we further refine our model for the site in this System Design phase of modeling.

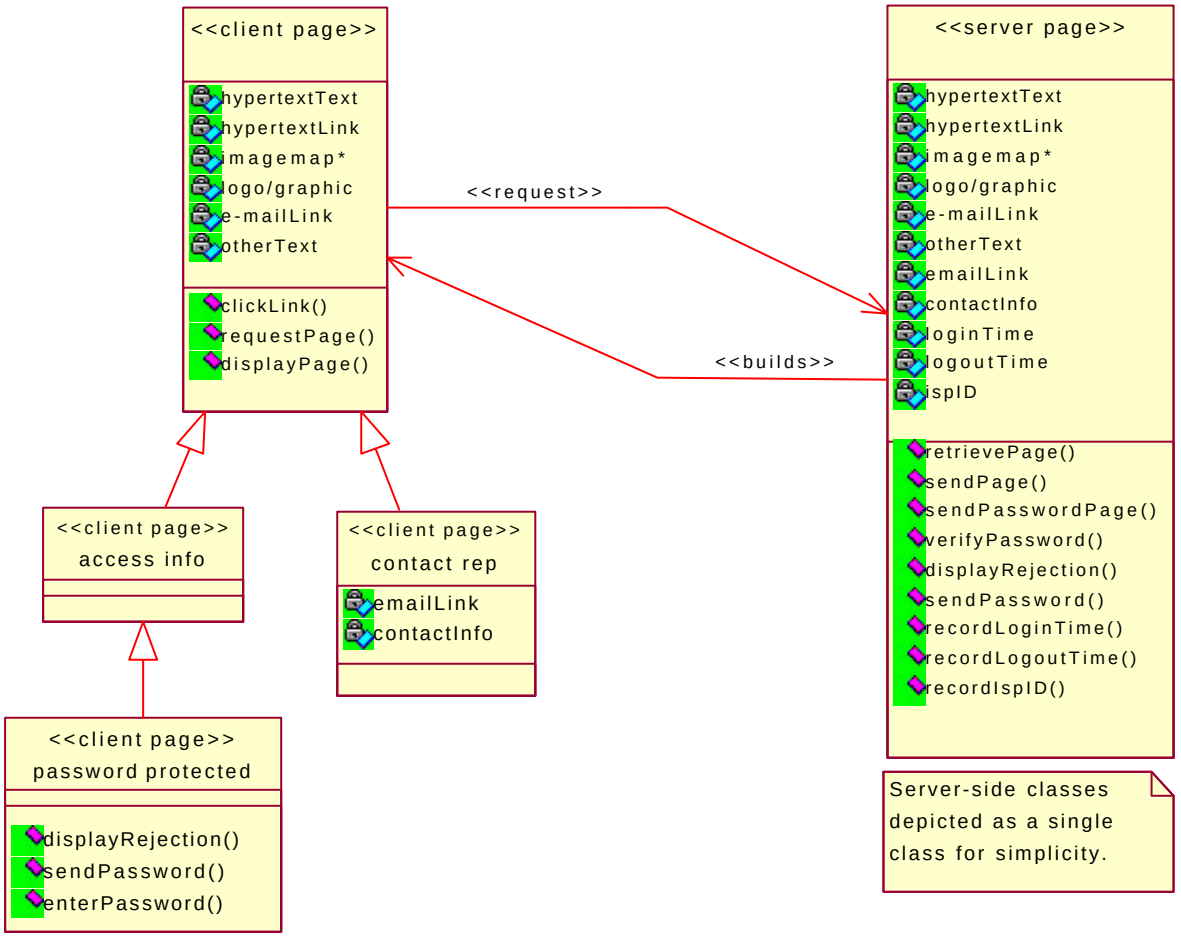


Figure 6: Design-Level Class Diagram

The operations included in our design-level Class Diagram were originally identified from the Sequence Diagram. We conclude our discussion of Class Diagrams with a list of the various UML elements as they map to the artifacts of a Web site.

Web Artifacts	UML Elements
User	Actor
HTML hyperlink*	Association element / control object
Hyperlink text	Entity object
Site Map	Component Diagram
Storyboard	Component Diagram
Server page	<<server page>>
Client page	<<client page>> / boundary object
Java Script	<<java script>>
HTML form	<<form>>
HTML target of a frame	<<target>>
HTML frameset	<<frameset>>
Various groups of related elements	Packages of these related elements

* The <<link>> association has a list of parameter names that are sent along with the link request. The server then processes this link request along with any parameters [Conallen, 1999]. Hyperlinks request a specific page, either within the site or a site stored on another computer that is accessible through the Web (i.e. through HTTP).

4. PHYSICAL DESIGN

4.1 Component Diagrams

An UML-based component diagram is essentially a site map, which provides an overview of client-side navigation through high-level abstraction of the various pages of the Web site. The components of a Component Diagram, as they relate to a Web site, include each of the pages and the hyperlinks (navigational links) among, and between, the pages [Conallen, 1999]. (Due to space limitations, we have not included a component diagram.) Components, however, only represent the "'physical' packaging" [Conallen, 1999]. As such, they provide no value when modeling any of the workings of the component, which are conceptually internal to the page [Conallen, 1999]. When considering a Web site, these internal workings could include inter-page links, scripts, Java applets and Active Server Pages (ASP). As we have attempted to demonstrate through this paper, other charts can be used to fill-in these details.

4.2 Deployment Diagrams

Deployment Diagrams provide a modeling mechanism for illustrating the physical components of a system. Figure 7 represents our conceptual representation of the physical components of a typical Web-based application.

It should be noted that the Application Server component is not necessary for our University Web site. It is included in the above figure only to illustrate a Web system that includes database functionality, which might be located on an Application Server. While a Deployment Diagram provides concise modeling for the physical structure of a Web site, it lends little, if any, insight into the development of our University site. As developers, we are primarily concerned with the workings of the site that directly interact with its end-users. We have presented several UML-based components that address these aspects of content presentation and navigational links.

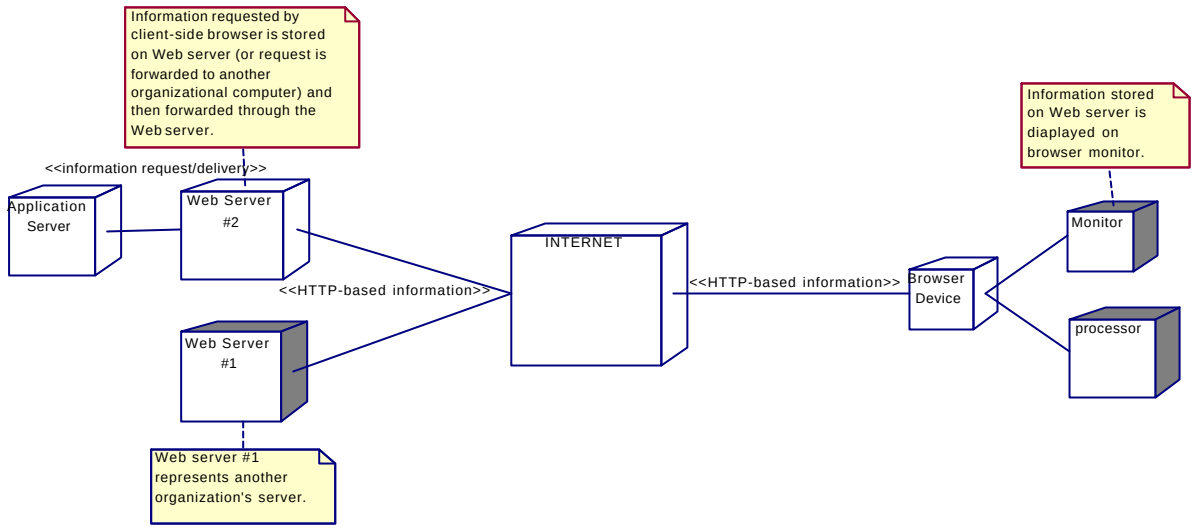


Figure 7: Deployment Diagram for a typical Web-based application

5. APPLICATIONS DESIGN

5.1 Interface Diagram

An Interface Diagram illustrates the navigation paths, similar to a Component Diagram. Directions of the various arrows indicate the navigational flow of control among, and between, the various pages. For clarity, we have deviated from software engineering notation in our Interface Diagram. Whereas standard UML notation includes the ability to depict bi-directional navigational flow (to and from a given hyperlink) through use of a single vertical line, each of our vertical lines is intended to depict a particular one-directional navigational flow. Figure 8 shows a partial Interface Diagram.

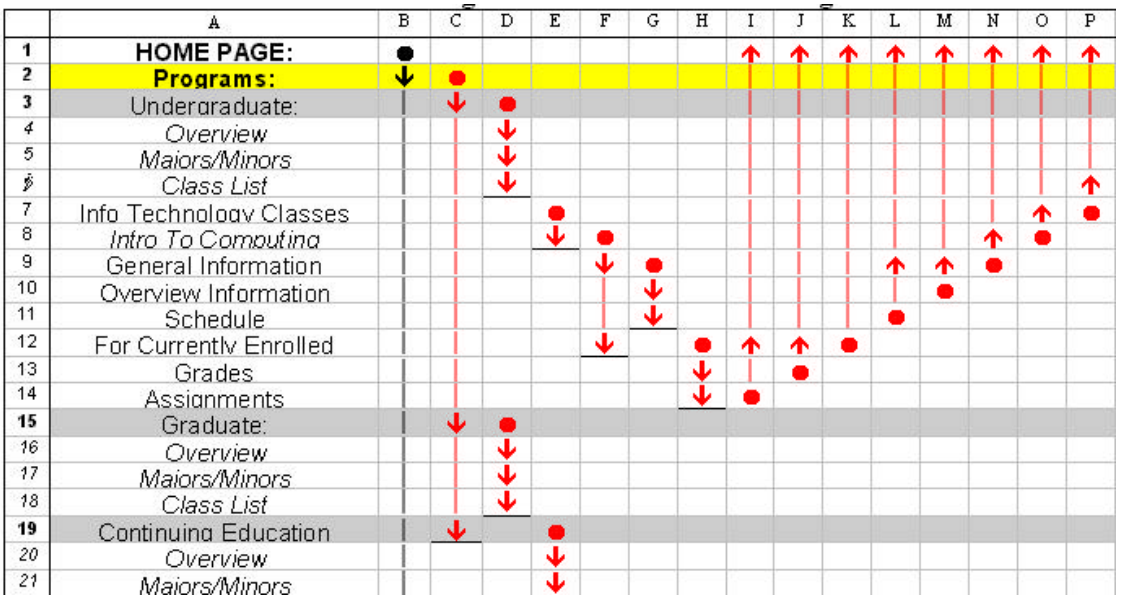


Figure 8: Partial Interface Diagram

6. Recommended Best Practices

We have compiled a list of some of the 'best practices' for UML-based modeling of Web sites.

General comments:

- Don't necessarily think that you need to develop every available UML diagram for every design effort you undertake.

- We have found that as when modeling any software development project, the level of detail of the various UML diagrams is determined by the relative sophistication of the particular Web site, as well as the particular focus of the developers of the site. You should attempt to model only at the level of abstraction that will be of the most value to you. This involves considerable judgment, and as such, requires experience.

- Don't waste too much time and effort on modeling the server side of a Web site, unless your particular site is intended to support transaction processing or other back-end processing functions for which the site will act as a front-end.

- When short on time, consider developing Use Case Descriptions, Class Diagrams and Sequence Diagrams, as Grady Booch suggested that 80% of the design effort can be accomplished through the development of these three tools.

- When severely short on time, develop at least an analysis-level (high-level) Class Diagram.

- Provide a numbering scheme for the Use Case Descriptions, Sequence Diagrams and Activity Diagrams (1, 1.x, 2, 2.x, etc.) and be consistent with this numbering scheme across each of the three diagrams. This will assist you and your end-users/clients in following the flow of processes and objects, which in turn assists in determining any oversights in your design.

- Consider developing the appropriate diagrams in the following order of priority:
  1. Problem Statement
  2. Use Case Diagrams
  3. Analysis-Level (high-level) Class Diagrams
  4. Sequence Diagrams
  5. State Diagrams
  6. Activity Diagrams
  7. Design-Level (low-level) Class Diagrams
  8. Component Diagrams
  9. Deployment Diagrams
  10. Interaction Diagrams

SystemAnalysis:

- Consider using Packages to simplify your concepts, particularly when dealing with several groups of Actors. Grouping them by function or some other common denominator will simplify your diagrams and your process modeling. (Figure 3)

System Design:

- Include the text from your Use Case Description down the left side of your Sequence Diagram. This will serve as an organizational tool as you development your Diagram, Because Rational Rose does not 'link' the steps of your Use Case Description to individual objects on your diagram, you will find yourself doing additional moving (vertically) of objects within your diagram if you add any additional objects/procedures. Therefore, in an effort to minimize the

number and complexity of your edits, we suggest that if you are using Rose as your design tool, draw a sketch, or two, of your Sequence Diagram on an oversized piece of paper prior to developing the Diagram in Rose. (Figure 5)

- Develop additional Sequence Diagrams for each alternative course of action that may need to be modeled as part of the Use Case. This will help maintain the clarity of your original Diagram.

- We believe you will gain the most benefit from the development of a State Diagram if the Web site you are modeling is intended to serve as the front-end for some type of transaction processing system. Otherwise, as was the case with our sample University Web site, a State Diagram will not add much value to the design process.

- If you plan to develop a Activity Diagram, consider developing one that is based on physical swimlanes (client browser and Web server, as well as any other application server), particularly a site that supports transaction processing or other back-end processing functions for which the site will act as a front-end.

Physical Design:

- Use a Component Diagram to develop a model of the navigational links of your site. This essentially represents a storyboard of the site.

Applications Design:

- Consider color-coding the various rows of an Interface Diagram in order to provide additional visual structure for the navigation paths (Figure 8).

- Consider using our suggested notation when constructing an Interface Diagram (Figure 8), depending on the level of complexity of the site you are modeling. As noted, our notation results in a wider Diagram, but we feel it is more readable than the standard notation.

- Consider supplementing the Interface Diagram with a text-based outline of the navigational links.

- Don't waste too much time and effort on a Deployment Diagram, as it adds little value to the design effort of a Web site, particularly a site that doesn't support transaction processing or other back-end processing functions for which the site will act as a front-end. (Figure 7)

7. Conclusions

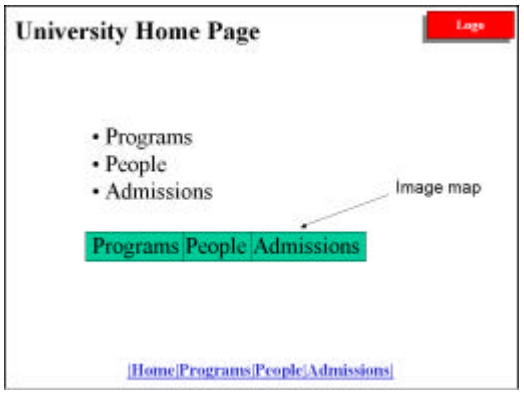
There is clearly a need for tools that are robust enough to assist developers in capturing the various views, constructs and capabilities of Web sites of varying complexity. Baresi and his colleagues [2000] referred to some of these complexities when they pointed out the various interrelationships between hypermedia design ("information structures and navigational paths") and functional design ("operations and applications behavior").

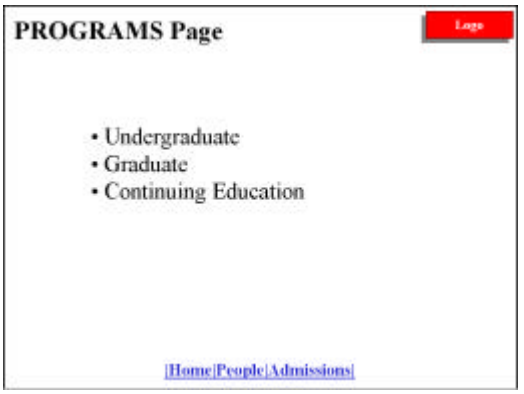
Through our personal experiences and research for this paper, as well as our development of the prototype University Web site, we found that the UML-based tools we have discussed combined to form a more robust and capable tool set than traditional methods of modeling sites (text-based descriptions, cognitive walkthroughs and storyboards). With some experience in utilizing these tools, designers can be

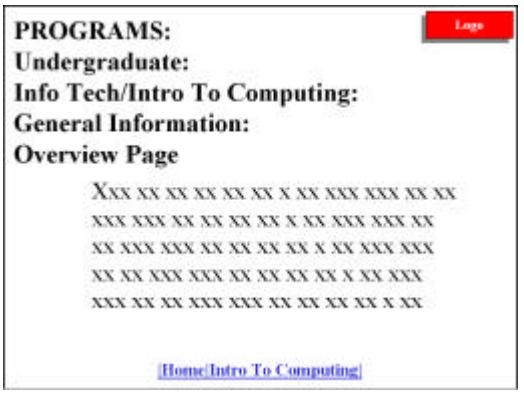
reasonably assured of developing a complete model of even complicated sites.

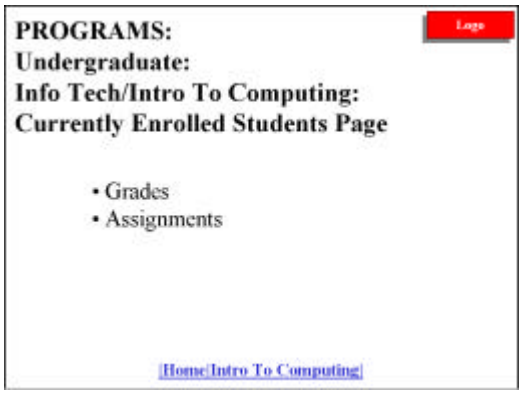
Lastly, we present the following thoughts regarding possible future research. We would be interested in investigating if there was any evidence to indicate whether or not using UML to model and develop a Web site increases the likelihood of creating a site that has a more useful, efficient and informative design than compared to a site that was developed utilizing traditional modeling techniques (text-based documentation, cognitive walkthroughs and storyboards). We believe it would also be both interesting and useful to investigate the relative usefulness and clarity of the various modifications we have introduced to some of our UML-based models. This could be accomplished through surveys and/or interviews with Web developers who would have had an opportunity to work with both standard-UML diagrams, as well as our modified diagrams.

Appendix A: Scope of University site (prototyped Web pages)



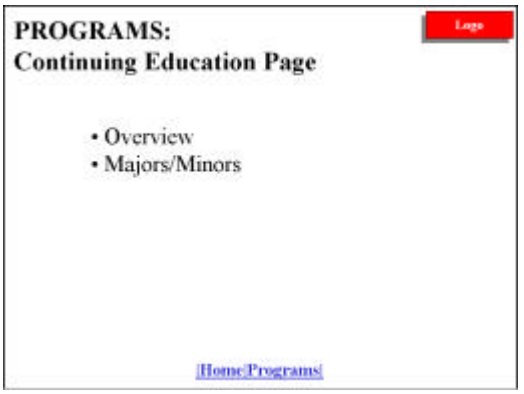


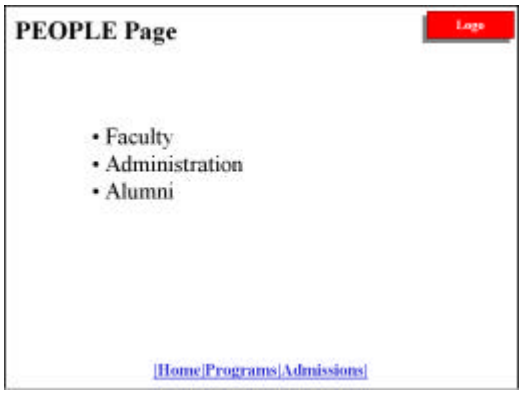



PROGRAMS:
Undergraduate:
Info Tech/Intro To Computing:
Currently Enrolled Students Page
• Grades
• Assignments
[Home|Intro To Computing]


PROGRAMS:
Graduate Page
• Overview
• Majors/Minors
• Class List
[Home|Programs]


PROGRAMS:
Graduate:
Class List Page
• Business Classes
• Education Classes
• Information Technology Classes
• Science & Math
[Home|Graduate]

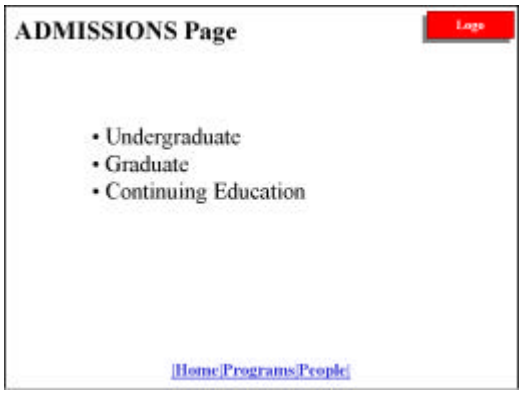

PROGRAMS:
Continuing Education Page
• Overview
• Majors/Minors
[Home|Programs]


PEOPLE Page
• Faculty
• Administration
• Alumni
[Home|Programs|Admissions]


PEOPLE:
Faculty Page
• Overview
• Faculty List
[Home|People]

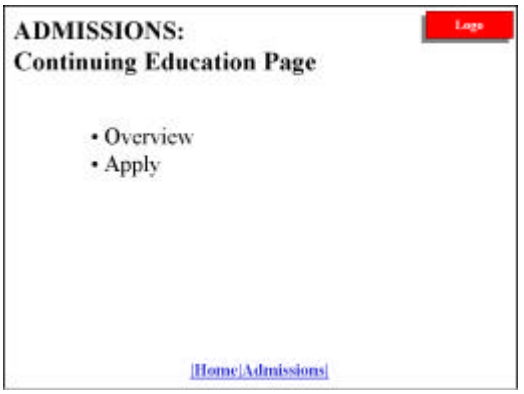

PEOPLE:
Administration Page
• Overview
• Administrator List
[Home|People]


PEOPLE:
Alumni Page
• Overview
• Alumni List
[Home|People]


ADMISSIONS Page
Logo
• Undergraduate
• Graduate
• Continuing Education
|Home|Programs|People|


ADMISSIONS:
Undergraduate Page
Logo
• Overview
• Apply
|Home|Admissions|


ADMISSIONS:
Graduate Page
Logo
• Overview
• Apply
|Home|Admissions|


ADMISSIONS:
Continuing Education Page
Logo
• Overview
• Apply
|Home|Admissions|

References

Baresi, L., Garzotto, F. and Paolini, P. "Conceptual Design of Web Applications With UML". *Technical Report. Politecnico di Milano* (April 2000).

Booch, Gary, James Rumbaugh, Ivar Jacobson. (1999) *The Unified Modeling Language User Guide*, Addison Wesley Longman, Inc., Massachusettes.

Brannan, James. "Modeling IIS Components With UML and Visual Modeler". *ASP Today* [online] http:/www.asptoday.com/articles/20000314.htm. (March 14, 2000) Accessed June 30, 2000.

Conallen, Jim. "Modeling Web Application Architecture With UML". *Communications of The ACM.* Volume 42, Number 10 (October 1999) p. 63-70.

Conallen, Jim. "Modeling ASP Applications With UML." *ASP Today* [online] http:/www.asptoday.com/articles/19990517.htm. (May 17, 1999) Accessed June 30, 2000.

Kalakota, Ravi and Andrew B. Whinston, *Electronic Commerce: A Manager's Guide*. Reading, Massachusetts. Addison Wesley Longman, Inc. 1997.

Parsons, June and Dan Oja. *Computer Concepts: Brief Edition.* Cambridge, Massachusetts. Course Technology. 1998.

Rosenberg, Doug. *Use Case Driven Object Modeling With UML*. Reading, Massachusetts. Addison Wesley Longman, Inc. 1999.

———