

provide a rather one-dimensional view of the interaction. Sequence diagrams present a second dimension that is more procedural (dynamic) in nature. State diagrams provide a third dimension that is more behavioral and contains information about potential navigation pathways that is not provided by use-cases or the sequence diagram. When all three dimensions are used, omissions or inconsistencies that might escape discovery in one dimension become obvious when a second (or third) dimension is examined. It is for this reason that large complex WebApps can benefit from an interaction model that encompasses all three representations.

User interface prototype. The layout of the user interface, the content it presents, the interaction mechanisms it implements, and the overall aesthetic of the user-WebApp connections have much to do with user satisfaction and the overall acceptance of the WebApp. Although it can be argued that the creation of a user interface prototype is a design activity, it is a good idea to perform it during the creation of the analysis model. The sooner that a physical representation of a user interface can be reviewed, the higher the likelihood that end-users will get what they want. User interface analysis and design are discussed in detail in Chapter 12.

Because WebApp development tools are plentiful, relatively inexpensive, and functionally powerful, it is best to create the interface prototype using such tools. The prototype should implement the major navigational links and represent the overall screen layout in much the same way that it will be constructed.



Some Web engineers prefer a pencil and paper storyboard of major WebApp pages (screens). Although such storyboards can be developed very quickly, the navigation flow is less obvious than with an operational prototype.

18.5 THE FUNCTIONAL MODEL

The *functional model* addresses two processing elements of the WebApp, each representing a different level of procedural abstraction: (1) user observable functionality that is delivered by the WebApp to end-users, and (2) the operations contained within analysis classes that implement behaviors associated with the class.

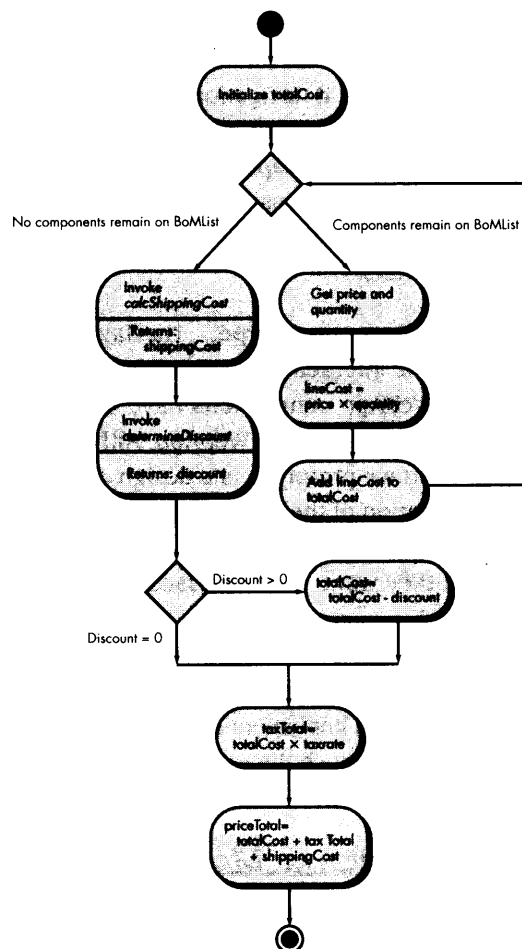
User-observable functionality encompasses any processing functions that are initiated directly by the user. For example, a financial Web site might implement a variety of financial functions (e.g., a college tuition savings calculator or a retirement savings calculator). These functions may actually be implemented using operations within analysis classes, but from the point of view of the end-user, the function (more correctly, the data provided by the function) is the visible outcome.

At a lower level of procedural abstraction, the analysis model describes the processing to be performed by analysis class operations. These operations manipulate class attributes and are involved as classes collaborate with one another to accomplish some required behavior.

Regardless of the level of procedural abstraction, the UML activity diagram can be used to represent processing details. Figure 18.7 depicts an activity diagram for the

FIGURE 18.7

Activity diagram for `computePrice()` operation



As an alternative, you can also write a simple processing narrative or program design language representation (Chapter 11). However, many people prefer a graphical representation.

`computePrice()` operation that is part of the **BillOfMaterials** analysis class.⁶ As we noted in Chapter 8, the activity diagram is similar to the flowchart, illustrating the processing flow and logical decisions with the flow. It should be noted that two additional operations are invoked within the procedural flow: `calcShippingCost()`, which calculates the cost of shipping depending upon the shipping method chosen by the customer, and `determineDiscount()`, which determines any special discounts for the *SafeHome* components that were selected for purchase. The construction details indicating how these operations are invoked and the interface details for each operation are not considered until WebApp design commences.

⁶ A review of the **BillOfMaterials** analysis class might determine that in the interest of cohesion, the `computePrice()` operation might best be placed within an **Invoice** class. This suggestion has merit. However, it remains within the **BillOfMaterials** analysis class for the purposes of this example.

18.6 THE CONFIGURATION MODEL

WebApps must be designed and implemented in a manner that accommodates a variety of environments on both the server-side and the client-side.⁷ The WebApp can reside on a server that provides access via the Internet, an Intranet, or an Extranet. Server hardware and operating system environment must be specified. In addition, interoperability considerations on the server-side should be considered. If the WebApp must access a large database or interoperate with corporate applications that exist on the server side, appropriate interfaces, communication protocols, and related collaborative information must be specified.



Although it's very important to consider all configurations that are likely to be used, remember that a WebApp must be engineered to serve its end-users, not the idiosyncrasies of a particular browser.

Client-side software provides the infrastructure that enables access to the WebApp from the user's location. In general, browser software is used to deliver the WebApp content and functionality that is downloaded from the server. Although standards do exist, each browser has its own peculiarities. For this reason, the WebApp must be thoroughly tested within every browser configuration that is specified as part of the *configuration model*.

In some cases, the configuration model is nothing more than a list of server-side and client-side attributes. However, for more complex WebApps, a variety of configuration complexities (e.g., distributing load among multiple servers, caching architectures, remote databases, multiple servers serving various objects on the same Web page) may have an impact on analysis and design. The UML deployment diagram (Chapter 10) can be used in situations in which complex configuration architectures must be considered.

18.7 RELATIONSHIP-NAVIGATION ANALYSIS

The elements of the analysis model described in the preceding sections identify content and functional elements along with the manner in which they are used to implement user interaction. As analysis evolves into design, these elements become part of the WebApp architecture. In the context of Web applications, each architectural element has the potential to be linked to all other architectural elements. But as the number of links increases, navigational complexity throughout the WebApp also increases. The question, then, is how to establish the appropriate links between content objects and among the functions that provide user-required capabilities.

"[Navigation] is not only the action of jumping from page to page, but the idea of moving through an information space."

A. Reina and J. Torres

⁷ The *server-side* hosts the WebApp and all related system features that enable multiple users to gain access to the WebApp via a network. The *client-side* provides a software environment (e.g., browsers) that enable end-users to interact with the WebApp on the user's desktop.

Relationship-navigation analysis (RNA) provides a series of analysis steps that strive to identify relationships among the elements uncovered as part of the creation of the analysis model.⁸ Yoo and Bieber [YOO00] describe RNA in the following manner:

RNA provides systems analysts with a systematic technique for determining the relationship structure of an application, helping them to discover all potentially useful relationships in application domains. These later may be implemented as links. RNA also helps determine appropriate navigational structures on top of these links. RNA enhances system developers' understanding of application domains by broadening and deepening their conceptual model of the domain. Developers can then enhance their implementation by including additional links, metainformation, and navigation.

The RNA approach is organized into five steps:


- *Stakeholder analysis*—identifies the various user categories (as described in Section 18.1) and establishes an appropriate stakeholder hierarchy.
- *Element analysis*—identifies the content objects and functional elements that are of interest to end-users (as described in Sections 18.3 and 18.5).
- *Relationship analysis*—describes the relationships that exist among the WebApp elements.
- *Navigation analysis*—examines how users might access individual elements or groups of elements.
- *Evaluation analysis*—considers pragmatic issues (e.g., cost/benefit) associated with implementing the relationships defined earlier.

The first two steps in the RNA approach have been discussed earlier in this chapter. In the sections that follow, we consider methods for establishing the relationships that exist among content objects and functions.

18.7.1 Relationship Analysis—Key Questions

Yoo and Bieder [YOO00] suggest a list of questions that a Web engineer or systems analyst should ask about each element (content object or function) that has been identified within the analysis model. The following list, adapted for WebApps, is representative [YOO00]:

- Is the element a member of a broader category of elements?
- What attributes or parameters have been identified for the element?
- Does descriptive information about the element already exist? If so, where is it?
- Does the element appear in different locations within the WebApp? If so, where?
- Is the element composed of other smaller elements? If so, what are they?

 **How do we assess analysis model elements to understand the relationships between them?**

⁸ It should be noted that RNA can be applied to any information system and was originally developed for hypermedia systems in general. It can, however, be adapted nicely for Web engineering.

- Is the element a member of a larger collection of elements? If so, what is it and what is its structure?
- Is the element described by an analysis class?
- Are other elements similar to the element being considered? If so, is it possible that they could be combined into one element?
- Is the element used in a specific ordering of other elements? Does its appearance depend on other elements?
- Does another element always follow the appearance of the element being considered?
- What pre- and post-conditions must be met for the element to be used?
- Do specific user categories use the element? Do different user categories use the element differently? If so, how?
- Can the element be associated with a specific formulation goal or objective? With a specific WebApp requirement?
- Does this element always appear at the same time as other elements appear? If so, what are the other elements?
- Does this element always appear in the same place (e.g., same location of the screen or page) as other elements? If so, what are the other elements?

The answers to these and other questions help the Web engineer to position the element in question within the WebApp and to establish relationships among elements.

It is possible to develop a relationship taxonomy and to categorize each relationship identified as a result of the questions noted. The interested reader should refer to [YOO00] for more detail.

18.7.2 Navigation Analysis

Once relationships have been developed among elements defined within the analysis model, the Web engineer must consider the requirements that dictate how each user category will navigate from one element (e.g., content object) to another. The mechanics of navigation are defined as part of design. At this stage, developers should consider overall navigation requirements. The following questions should be asked and answered:

What questions should be asked to better understand navigation requirements?

- Should certain elements be easier to reach (require fewer navigation steps) than others? What is the priority for presentation?
- Should certain elements be emphasized to force users to navigate in their direction?
- How should navigation errors be handled?
- Should navigation to related groups of elements be given priority over navigation to a specific element?

Select a WebApp that you visit regularly from one of the following categories: (a) news or sports, (b) entertainment, (c) e-commerce, (d) gaming, (e) computer-related, (f) a WebApp recommended by your instructor. Perform the activities noted in Problems 18.6 through 18.12:

- 18.6.** Develop one or more use-cases that describe specific user behavior for the WebApp.
- 18.7.** Select a content object or function that is part of the WebApp architecture and answer the relationship-navigation questions listed in Section 18.7.1.
- 18.8.** Develop a UML sequence diagram and a UML state diagram that describes a specific interaction within the WebApp.
- 18.9.** Consider the existing WebApp interface. Prototype a change to the interface that you believe will improve it.
- 18.10.** Considering the existing WebApp, answer the relationship-navigation questions listed in Section 18.7.2.
- 18.11.** Represent a partial content hierarchy and define at least three analysis classes for the WebApp.
- 18.12.** Select a user observable function provided by the WebApp and model it using a UML activity diagram.

FURTHER READINGS AND INFORMATION SOURCES

Many books dedicated to analysis modeling for conventional software—with particular emphasis on use-cases and UML notation—contain much useful information that can be readily adapted by Web engineers. Use-cases form the foundation of analysis modeling for WebApps. Books by Kulak and his colleagues (*Use Cases: Requirements in Context*, second edition, Addison-Wesley, 2004), Bittner and Spence (*Use Case Modeling*, Addison-Wesley, 2002), Cockburn (*Writing Effective Use Cases*, Addison-Wesley, 2001), Armour and Miller (*Advanced Use-Case Modeling: Software Systems*, Addison-Wesley, 2000), Rosenberg and Scott (*Use Case Driven Object Modeling with UML: A Practical Approach*, Addison-Wesley, 1999), and Schneider, Winters, and Jacobson (*Applying Use Cases: A Practical Guide*, Addison-Wesley, 1998) provide worthwhile guidance in the creation and use of this important requirements representation mechanism. Worthwhile discussions of UML have been written by Arlow and Neustadt (*UML and the Unified Process*, Addison-Wesley, 2002), Schmuller (*Teach Yourself UML*, Sams Publishing, 2002), Booch and his colleagues (*The UML User Guide*, Addison-Wesley, 1998), and Rumbaugh and his colleagues (*The Unified Modeling Language Reference Manual*, Addison-Wesley, 1998).

Books dedicated to Web site design often contain one or two chapters that discuss analysis issues (although these are often cursory discussions). The following books contain one or more aspects of analysis within the context of Web engineering: Van Duyne and his colleagues (*The Design of Sites*, Addison-Wesley, 2002), Rosenfeld and Morville (*Information Architecture for the World Wide Web*, O'Reilly & Associates, 2002), Wodtke (*Information Architecture*, New Riders Publishing, 2002), Garrett (*The Elements of User Experience: User Centered Design for the Web*, New Riders Publishing, 2002), Niederst (*Web Design in a Nutshell*, O'Reilly & Associates, 2001), Lowe and Hall (*Hypertext and the Web: An Engineering Approach*, Wiley, 1999), and Powell (*Web Site Engineering*, Prentice-Hall, 1998) provide reasonably complete coverage. Norris, West, and Watson (*Media Engineering: A Guide to Developing Information Products*, Wiley, 1997), Navarro and Khan (*Effective Web Design: Master the Essentials*, Sybex, 1998), and Fleming and Koman (*Web Navigation: Designing the User Experience*, O'Reilly & Associates, 1998) provide additional guidance for analysis and design.

A wide variety of information sources on analysis modeling for Web engineering is available on the Internet. An up-to-date list of World Wide Web references can be found under “software engineering resources” at the SEPA Web site:

<http://www.mhhe.com/pressman>.

**KEY
CONCEPTS**

aesthetic design
 architecture design
 component-level design
 content architecture
 content design
 interface design
 MVC architecture
 navigation design
 OOHDM
 metrics
 patterns
 quality attributes

In his authoritative book on Web design, Jakob Nielsen [NIE00] states: “There are essentially two basic approaches to design: the artistic ideal of expressing yourself and the engineering ideal of solving a problem for a customer.” During the first decade of Web development, the artistic idea was the approach that many developers chose. Design occurred in an ad hoc manner and was usually conducted as HTML was generated. Design evolved out of an artistic vision that itself evolved as WebApp construction occurred.

Even today, the most “extreme” proponents of agile software development (Chapter 4) use Web applications as poster children for “limited design.” They argue that WebApp immediacy and volatility mitigate against formal design, that design evolves as an application is built (coded), and that relatively little time should be spent on creating a detailed design model. This argument has merit, but only for relatively simple WebApps. When content and function are complex; when the size of the WebApp encompasses hundreds of content objects, functions, and analysis classes; when the success of the WebApp will have a direct impact on the success of the business, design cannot and should not be taken lightly.

This reality leads us to Nielsen’s second approach—“the engineering ideal of solving a problem for a customer.” Web engineering adopts this philosophy, and a more rigorous approach to WebApp design enables developers to achieve it.

**QUICK
LOOK**

What is it? Design for WebApps encompasses technical and non-technical activities. The look and feel of content is developed as part of graphic design; the aesthetic layout of the user interface is created as part of interface design; and the technical structure of the WebApp is modeled as part of architectural and navigational design. In every instance, a design model should be created before construction begins, but a good Web engineer recognizes that the design will evolve as more is learned about stakeholder requirements as the WebApp is built.

Who does it? Web engineers, graphic designers, content developers, and other stakeholders all participate in the creation of a design model for Web engineering.

Why is it important? Design allows a Web engineer to create a model that can be assessed for quality and improved before content and code are generated, tests are conducted, and end users become involved in large numbers. Design is the place where WebApp quality is established.

What are the steps? WebApp design encompasses six major steps that are driven by information obtained during analysis modeling.

Design Model. Design models are designed to be used as a guide for establishing the content, structure, and look of the WebApp. Aesthetic design (also called graphic design) establishes the look and feel of the end-user sees. Architectural design focuses on the overall hypermedia structure of all content objects and functions. Interface design establishes the overall layout and interaction mechanisms that define the user interface. Navigation design defines how the end-user navigates through the hypermedia structure, and component design represents the detailed internal structure of functional elements of the WebApp.

What is the work product? A design model that encompasses content, aesthetics, architecture, interface, navigation, and component-level design issues is the primary work product of Web engineering design.

How do I ensure that I've done it right?

Each element of the design model is reviewed by the Web engineering team (and selected stakeholders) in an effort to uncover errors, inconsistencies, or omissions. In addition, alternatives are considered, and the degree to which the current design model will lead to an effective implementation is also reviewed.

19.1 DESIGN ISSUES FOR WEB ENGINEERING

When design is applied within the context of Web engineering, both generic and specific issues must be considered. From a generic viewpoint, design results in a model that guides the construction of the WebApp. The design model, regardless of its form, should contain enough information to reflect how stakeholder requirements (defined in an analysis model) are to be translated into content and executable code. But design must also be specific. It must address key attributes of a WebApp in a manner that enables a Web engineer to build and test effectively.

19.1.1 Design and WebApp Quality

In earlier chapters, we noted that design is the engineering activity that leads to a high-quality product. This leads us to a recurring question that is encountered in all engineering disciplines: What is quality? In this section we examine the answer within the context of Web engineering.

Every person who has surfed the Web or used a corporate Intranet has an opinion about what makes a "good" WebApp. Individual viewpoints vary widely. Some users enjoy flashy graphics, others want simple text. Some demand copious information, others desire an abbreviated presentation. Some like sophisticated analytical tools or database access, others like to keep it simple. In fact, the user's perception of "goodness" (and the resultant acceptance or rejection of the WebApp as a consequence) might be more important than any technical discussion of WebApp quality.

But how is WebApp quality perceived? What attributes must be exhibited to achieve goodness in the eyes of end-users and at the same time exhibit the technical characteristics of quality that will enable a Web engineer to correct, adapt, enhance, and support the application over the long term?