Vinod: You were telling me about the user's objectives and background.

Marketing person #1: Like I said, we want the user to be able to customize the entire *SafeHome* system, you know, pick sensors, control panels, features and functions, then get a "bill of materials" automatically generated, get pricing, and then purchase the system via the Web site.

Marketing person #2: We assume that the user is a homeowner—not technical—so we need to guide him or her through the process step by step.

Marketing person #3: I'm not technical, but I'm worried about the specialty stuff that we need to do in addition to the basic e-commerce stuff.

Vinod (addressing #3): Meaning?

Marketing person #3: The hard part is going to be guiding the user through the "customizing process" in a way that is simple and complete. The actual e-commerce part is pretty straightforward.

Marketing person #1: We've got to provide an 800 number for people who don't want to do the customization themselves.

Marketing person #3: I agree.

Vinod: Okay, we're going to have to talk about exactly how you'd like to do the product customization as a [illegible] activity, but let's hold on that for a moment. I have a few other fundamental questions.

Vinod (looking at Marketing person #2): You said that you wanted to guide the users through the [illegible] process. Any special approach?

Marketing person #2: I'd like to see a step-by-step process, with fill-in-the-blanks responses to basic requirements questions, pull down menus, that sort of thing. Each step is a window, and each window's data is validated before moving to the next step.

Vinod: Have you checked that out with representative users?

Marketing person #2: No, but I will.

Vinod: One more thing . . . how does a user find our site?

Marketing person #1: We're working on an ad campaign that will paste www.SafeHomeAssured.com in magazine ads, targeted direct mail, [illegible] that appear in search engines, and maybe [illegible] and radio spots.

Vinod: What I mean is . . . they'll always enter through the home page.

Marketing person #3: That's what we'd like.

Vinod: Okay, now we've got to get to work. Let's explore the details of how you want to customize [illegible] on-line.

Communicating with stakeholders and end-users. Most WebApps have a broad population of end-users. Although the creation of user categories or classes makes an evaluation of user requirements more manageable, it is not advisable to use information gathered from just one or two people as the basis for formulation or analysis. More people (and more opinions/points of view) must be considered.

Communication can be accomplished using one or more of the following mechanisms [FUC02a]:

What communication mechanisms can be used in WebE work?

- *Traditional focus groups*—a trained moderator meets with a small (usually fewer than 10 people) group of representative end-users (or internal stakeholders playing the role of end-users). The intent is to discuss the WebApp to be developed, and, out of the discussion, to better understand requirements for the system.
- *Electronic focus groups*—a moderated electronic discussion conducted with a group of representative end-users and stakeholders. The number of people who participate can be larger. Because all users can participate at the same

time, more information can be collected in a shorter time period. Since all discussion is text-based, a contemporaneous record is automatic.

- *Iterative surveys*—a series of brief surveys, addressed to representative users and requesting answers to specific questions about the WebApp are conducted via a Web site or e-mail. Responses are analyzed and used to fine-tune the next survey.
- *Exploratory surveys*—a Web-based survey that is tied to one or more WebApps that have users who are similar to the ones that will use the WebApp to be developed. Users link to the survey and respond to a series of questions (usually receiving some reward for participation).
- *Scenario-building*—selected users are asked to create informal use-cases that describe specific interactions with the WebApp.

ADVICE

An evaluation of content objects and operations can be delayed until analysis modeling begins. At this point it's more important to collect information, not evaluate it.

Analyzing information gathered. As information is gathered, it is categorized by user class and transaction type, and then assessed for relevance. The objective is to develop lists of content objects, operations that are applied to content objects within a specific user transaction, functions (e.g., informational, computational, logical, and help-oriented) that the WebApp provides for end-users, and other nonfunctional requirements that are noted during the communication activities.

Fuccella and Pizzolato [FUC02b] suggest a simple (low-tech) method for understanding how content and functionality should be organized. A stack of "cards" is created for content objects, operations applied to content objects, WebApp functions, and other nonfunctional requirements. The cards are shuffled into random order and then distributed to representatives from each user category. The users are asked to arrange the cards into groupings that reflect how they would like content and functionality to be organized within the WebApp. Users are then asked to describe each grouping and the reasons why it is important to them. Once each user performs this exercise, the Web engineering team looks for common groupings among different user classes and other groupings that are unique to a specific user class.

The WebE team develops a list of labels that would be used to point to information within each of the groupings derived using the card stacks. Different representative users are then given the card stacks and asked to allocate content and functionality to each of the labels. The intent here is to determine when the labels (links within the actual WebApp) properly imply access to content and functions that the users expect to find behind the label. This step is applied iteratively until consensus is achieved.

"If you cannot describe what you are doing as a process, you don't know what you're doing."

W. E. Deming

ADVICE

Use-cases have been discussed in detail in Part 2 of this book. Although many advocate the development of lengthy use-cases, even an informal narrative provides some benefit. Convince users to write use-cases.

Developing use-cases. Use-cases[2] describe how a specific user category (called an *actor*) will interact with the WebApp to accomplish a specific action. The action may be as simple as acquiring defined content, or as complex as conducting detailed user-guided analysis of selected records maintained in an on-line database. The use-case describes the interaction from the user's point of view.

Although developing and analyzing them takes time, use-cases (1) help the developer to understand how users perceive their interaction with the WebApp; (2) provide the detail necessary to create an effective analysis model; (3) help compartmentalize WebE work; and (4) provide important guidance for those who must test the WebApp.

TASK SET

Customer Communication (Analysis/Formulation)

1. *Identify business stakeholders.* Exactly who is the "customer" for the WebApp? What business people can serve as experts and representative end-users? Who will serve as an active member of the team?
2. *Formulate the business context.* How does the WebApp fit into a broader business strategy?
3. *Define key business goals and objectives for the WebApp.* How is the success of the WebApp to be measured in both qualitative and quantitative terms?
4. *Define informational and applicative goals.* What classes of content are to be provided to end-users? What functions/tasks are to be accomplished when using the WebApp?
5. *Identify the problem.* What specific problem does the WebApp solve?
6. *Gather requirements.* What user tasks will be accomplished using the WebApp? What content is to be developed? What interaction metaphor will be used? What computational functions will be provided by the WebApp? How will the WebApp be configured for network utilization? What navigation scheme is desired?

ADVICE

For small projects, a simple "requirements database" may be maintained (using a spreadsheet) in lieu of UML models. This allows all members of a WebE team to trace requirements to the content and function delivered and to better control the inevitable stream of changes that will occur.

17.1.3 The Bridge to Analysis Modeling

As we have noted earlier in this chapter, the activities that lead a Web engineering team from formulation to analysis modeling represent a continuum. In essence, the level of abstraction considered during the early stages of formulation is business strategic. However, as formulation proceeds, tactical details are discussed and specific WebApp requirements are addressed. Ultimately, these requirements are modeled (using use-cases and UML notation).

The concepts and principles discussed for software requirements analysis (Chapters 7 and 8) apply without revision for the Web engineering analysis activity. During analysis, scope defined during the formulation activity is elaborated to create a complete analysis model for the WebApp. Four different types of analysis are conducted during WebE: content analysis, interaction analysis, function analysis and configuration analysis. Each of these analysis tasks and the modeling techniques associated with them is discussed in Chapter 18.

2 Techniques for developing use-cases have been presented in detail in Chapters 7 and 8.

"By failing to prepare, you are preparing to fail."

Benjamin Franklin

17.2 Planning for Web Engineering Projects

Given the immediacy of WebApps, it is reasonable to ask: Do we really need to spend time planning and managing a WebApp effort? Shouldn't we just let a WebApp evolve naturally, with little or no explicit management? More than a few Web developers would opt for little or no management, but that doesn't make them right!

WebRef
Tools that assist an e-project manager can be found at **www.eproject.com.**

Figure 17.1 presents a table adapted from Kulik and Samuelsen [KUL00] that indicates how "e-projects" (their term for WebApp projects) compare to traditional software projects. Referring to the figure, traditional software projects and major e-projects have substantial similarities. Since project management is indicated for traditional projects, it would seem reasonable to argue that it would also be indicated for major e-projects. Small e-projects do have special characteristics that make them different from traditional projects. However, even in the case of small e-projects, planning must occur, risks must be considered, a schedule must be established, and controls must be defined so that confusion, frustration, and failure are avoided.

FIGURE 17.1 Differences between traditional projects and e-projects [adapted from KUL00]

	Traditional projects	Small e-Projects	Major e-Projects
Requirements gathering	Rigorous	Limited	Rigorous
Technical specifications	Robust: models, spec	Descriptive overview	Robust: UML models, spec
Project duration	Measured in months or years	Measured in days, weeks or months	Measured in months or years
Testing and QA	Focused on achieving quality targets	Focused on risk control	SQA as described in Chapter 26
Risk management	Explicit	Inherent	Explicit
Half-life of deliverables	18 months or longer	3 to 6 months or shorter	6 to 12 months or shorter
Release process	Rigorous	Expedited	Rigorous
Post-release customer feedback	Requires proactive effort	Automatically obtained from user interaction	Obtained both automatically and via solicited feedback

17.3 The Web Engineering Team

A successful Web engineering team melds a wide variety of talents who must work as a team in a high-pressure project environment. Timelines are short, changes are relentless, and the technology keeps shifting. Creating a team that jells (see Chapter 21) is no simple matter.

"In today's net-centric and Web-enabled world, one now needs to know a lot about a lot."
Scott Tilley and Shihong Huang

17.3.1 The Players

The creation of a successful Web application demands a broad array of skills. Tilley and Huang [TIL99] address this issue when they state: "There are so many different aspects to [Web] application software that there is a (re)emergence of the renaissance person, one who is comfortable operating in several disciplines. . . ." While the authors are absolutely correct, "renaissance" people are in relatively short supply, and given the demands associated with major WebApp development projects, the diverse skill set required might be better distributed over a Web engineering team.

Web engineering teams can be organized in much the same way as traditional software teams (Chapter 21). However, the players and their roles are often quite different. Among the many skills that must be distributed across WebE team members are component-based software engineering, networking, architectural and navigational design, Internet standards/languages, human interface design, graphic design, content layout, and WebApp testing.

The following roles[3] should be distributed among the members of the WebE team:

What roles do people play on a WebE team?

Content developers/providers. Because WebApps are inherently content-driven, one role on the WebE team must focus on the generation and/or collection of content. Recalling that content spans a broad array of data objects, content developers/providers may come from diverse (nonsoftware) backgrounds.

Web publisher. The diverse content generated by content developers/providers must be organized for inclusion within the WebApp. In addition, someone must act as liaison between technical staff who engineer the WebApp and nontechnical content developers/providers. This role is filled by the *Web publisher*, who must understand both content and WebApp technology.

Web engineer. A Web engineer becomes involved in a wide range of activities during the development of a WebApp including requirements elicitation, analysis

3 These roles have been adapted from Hansen and his colleagues [HAN99].

modeling, architectural, navigational and interface design, WebApp implementation, and testing. The Web engineer should also have a solid understanding of component technologies, client/server architectures, HTML/XML, and database technologies and a working knowledge of multimedia concepts, hardware/software platforms, network security, and Web-site support issues.

Business domain experts. A business domain expert should be able to answer all questions related to the business goals, objectives and requirements associated with the WebApp.

Support specialist. This role is assigned to the person (people) who have responsibility for continuing WebApp support. Because WebApps continuously evolve, the support specialist is responsible for corrections, adaptations, and enhancements to the site, including updates to content, implementation of new procedures and forms, and changes to the navigation pattern.

Administrator. Often called the "Web Master," this person has responsibility for the day-to-day operation of the WebApp including: development and implementation of policies for the operation of the WebApp, establishment of support and feedback procedures, implementation of security and access rights, measurement and analysis of Web-site traffic, coordination of change control procedures (Chapter 27), and coordination with support specialists. The administrator may also be involved in the technical activities performed by Web engineers and support specialists.

17.3.2 Building the Team

In Chapter 21, guidelines for building successful software engineering teams are discussed in some detail. But do these guidelines apply in the pressure-packed world of WebApp projects? The answer is yes.

ADVICE

These characteristics are typical of collaborative, self-organizing teams that have adopted an agile philosophy (Chapter 4). The better your team, the better the software product you produce.

In his best selling book on a computer industry long past, Tracy Kidder [KID00] tells the story of a computer company's heroic attempt to build a computer to meet the challenge of a new product built by a larger competitor.[4] The story is a metaphor for teamwork, leadership, and the grinding stress that all technologists encounter when critical projects don't go as smoothly as planned.

A summary of Kidder's book hardly does it justice, but these key points [PIC01] have particular relevance when an organization builds a Web engineering team:

A set of team guidelines should be established. These encompass what is expected of each person, how problems are to be dealt with, and what mechanisms exist for improving the effectiveness of the team as the project proceeds.

4 Kidder's *The Soul of a New Machine,* originally published in 1981, is highly recommended reading for anyone who intends to make computing a career and everyone who already has!

Strong leadership is a must. The team leader must lead by example and by contact. She must exhibit a level of enthusiasm that gets other team members to "sign up" psychologically to the work that confronts them.

Respect for individual talents is critical. Not everyone is good at everything. The best teams make use of individual strengths. The best team leaders allow individuals the freedom to run with a good idea.

Every member of the team should commit. The main protagonist in Kidder's book calls this "signing up."

It's easy to get started, but it's very hard to sustain momentum. The best teams never let an "insurmountable" problem stop them. Team members develop a "good enough" solution and proceed, hoping that the momentum of forward progress may lead to an even better solution in the long term.

17.4 Project Management Issues for Web Engineering

Once formulation has occurred and basic WebApp requirements have been identified, a business must choose from one of two Web engineering options: (1) the WebApp is *outsourced*—Web engineering is performed by a third party vendor who has the expertise, talent, and resources that may be lacking within the business, or (2) the WebApp is developed *in-house* using Web engineers that are employed by the business. A third alternative, doing some Web engineering work in-house and outsourcing other work is also an option.

"As Thomas Hobbs observed in the 17th century, life under mob rule is solitary, poor, nasty, brutish, and short. Life on a poorly run software project is solitary, poor, nasty, brutish, and hardly ever short enough."

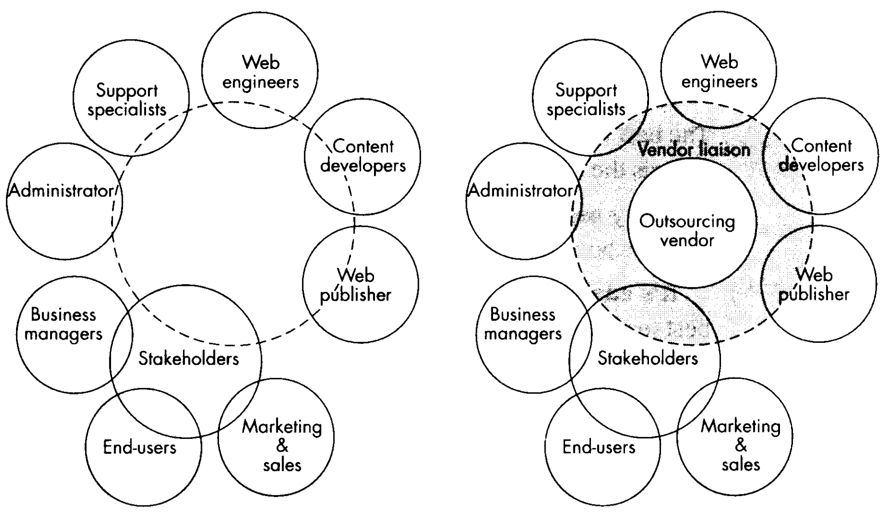
Steve McConnell

The work to be performed remains the same regardless of whether a WebApp is outsourced, developed in-house, or distributed between an outside vendor and in-house staff. But the communication requirements, the distribution of technical activities, the degree of interaction among stakeholders and developers, and a myriad of other critically important issues do change.

Figure 17.2 illustrates the organizational difference between outsourcing and in-house development for WebApps. In-house development (Figure 17.2a) integrates (the dashed circle implies integration) all members of the Web engineering team directly. Communication occurs using normal organizational pathways. For outsourcing (Figure 17.2b), it is both impractical and inadvisable to have each in-house constituency (e.g., content developers, stakeholders, internal Web engineers) communicate directly with the outsourcing vendor without some vendor liaison to coordinate and control communication. In the sections that follow, we examine planning for outsourcing and in-house development in more detail.

FIGURE 17.2

The organizational difference between outsourcing and in-house development



(a) in-house development (b) outsourced development

ADVICE

Do not assume that because you've outsourced a WebApp, your responsibilities are minimal. In fact, it's likely that more oversight and management, not less, will be required.

17.4.1 WebApp Planning—Outsourcing

A substantial percentage of WebApps are outsourced to vendors who (purportedly) specialize in the development of Web-based systems and applications.[5] In such cases, a business (the customer) asks for a fixed price quote for WebApp development from two or more vendors, evaluates competing quotes, and then selects a vendor to do the work. But what does the contracting organization look for? How is the competence of a WebApp vendor determined? How does one know whether a price quote is reasonable? What degree of planning, scheduling, and risk assessment can be expected as an organization (and its outsourcing contractor) embarks on a major WebApp development effort?

"Many Fortune 500 enterprises have discovered the software as a service model [outsourcing] and are employing similar models internally or externally."

Nick Evans

These questions are not always easy to answer, but a few guidelines are worth considering.

5 Although reliable industry data are difficult to find, it is safe to say that this percentage is considerably higher than the one encountered in conventional software work. Additional discussion of outsourcing may be found in Chapter 23.

Initiate the project. If outsourcing is the strategy to be chosen for WebApp development, an organization must perform a number of tasks before searching for an outsourcing vendor to do the work:

1. *Many of the analysis tasks discussed in Section 17.1.3 (and Chapter 18) should be performed internally.* The audience for the WebApp is identified; internal stakeholders who may have interest in the WebApp are listed; the overall goals for the WebApp are defined and reviewed; the information/services to be delivered by the WebApp are specified; competing Web sites are noted; and qualitative and quantitative "measures" of a successful WebApp are identified. This information should be documented in a product specification that is provided to the outsourcing vendor.

ADVICE

Some people argue that "rough design" is unnecessary. Look at it as a "first offer" that the outsourcing vendor can modify and improve upon. At least you're communicating your ideas about what the end-result should look like.

2. *A rough design for the WebApp should be developed internally.* Obviously, an expert Web developer will create a complete design, but time and cost can be saved if the general look and feel of the WebApp is identified for the outsourcing vendor (this can always be modified during preliminary stages of the project). The design should include an indication of the type and volume of content to be presented by the WebApp and the types of interactive processing (e.g., forms, order entry) to be performed. This information should be added to the product specification.
3. *A rough project schedule, including not only final delivery dates, but also milestone dates should be developed.* Milestones should be attached to deliverable versions (increments) of the WebApp as it evolves.
4. *A list of responsibilities for the internal organization and the outsourcing vendor is created.* In essence, this task addresses what information, contacts, and other resources are required of both organizations.
5. *The degree of oversight and interaction by the contracting organization with the vendor should be identified.* This should include the naming of a vendor liaison and the identification of the liaison's responsibilities and authority, the definition of quality review points as development proceeds, and the vendor's responsibilities with respect to interorganizational communication.

All of the information developed during these steps should be organized into a request for quote that is transmitted to candidate vendors.[6]

What guidelines should we use when considering various outsourcing vendors?

Select candidate outsourcing vendors. In recent years, thousands of "Web design" companies have emerged to help businesses establish a Web presence and/or engage in e-commerce. Many have become adept at the WebE process, but many others are little more than hackers. In order to select candidate Web developers, the

6 If WebApp development work is to be conducted by an internal group, nothing changes! The project is initiated in the same manner.

contractor must perform due diligence: (1) interview past clients to determine the Web vendor's professionalism, ability to meet schedule and cost commitments, and ability to communicate effectively; (2) determine the name of the vendor's chief Web engineer(s) for successful past projects (and later, be certain that this person is contractually obligated to be involved in your project); and (3) carefully examine samples of the vendor's work that are similar in look and feel (and business area) to the WebApp that is to be contracted. Even before a request for quote is offered, a face-to-face meeting may provide substantial insight into the "fit" between contractor and vendor.

"You pay peanuts, you get monkeys."
George Peppard playing Col. John "Hannibal" Smith on *The A-Team* (a 1980s TV show)

Assess the validity of price quotes and the reliability of estimates. Because relatively little historical data exist and the scope of WebApps is notoriously fluid, estimation is inherently risky. For this reason, some vendors will embed substantial safety margins into the cost quoted for a project. This is both understandable and appropriate. The question is *not* have we gotten the best bang for our buck. Rather, the questions should be:

- Does the quoted cost of the WebApp provide a direct or indirect return-on-investment that justifies the project?
- Does the vendor that has provided the quote exhibit the professionalism and experience we require?

If the answers to these questions are yes, the price quote is fair.

Understand the degree of project management you can expect/perform. The formality associated with project management tasks (performed by both the vendor and the contracting organization) is directly proportional to the size, cost, and complexity of the WebApp. For large, complex projects, a detailed project schedule that defines work tasks, SQA checkpoints, engineering work products, customer review points, and major milestones should be developed. The vendor and contractor should assess risk jointly and develop plans for mitigating, monitoring, and managing those risks that are deemed important. Mechanisms for quality assurance and change control should be explicitly defined in writing. Methods for effective communication between the contractor and the vendor should be established.

Assess the development schedule. Because WebApp development schedules span a relatively short period of time (often less than one or two months per delivered increment), the development schedule should have a fine granularity. That is, work tasks and minor milestones should be scheduled on a daily timeline. This fine granularity allows both the contracting organization and the vendor to recognize schedule slippage before it threatens the final completion date.

KEY POINT
To manage scope, the work to be performed within an increment is frozen. Changes are delayed until the next WebApp increment.

Manage scope. Because it is highly likely that scope will change as a WebApp project moves forward, the WebE process model is adaptable and incremental. This allows the vendor's development team to "freeze" scope for one increment so that an operational WebApp release can be created. The next increment may address scope changes suggested by a review of the preceding increment, but once the second increment commences, scope is again "frozen" temporarily. This approach enables the WebApp team to work without having to accommodate a continual stream of changes, but still recognizes the continuous evolution characteristic of most WebApps.

The guidelines suggested above are not intended to be a foolproof cookbook for the production of low-cost, on-time WebApps. However, they will help both the contracting organization and the vendor initiate work smoothly with a minimum of misunderstandings.

SAFEHOME

Outsourcing Preliminaries

The scene: Doug Miller's office.

The players: Doug Miller (manager of the *SafeHome* software engineering team) and Sharon Woods—an employee of e-CommerceSystems, the outsourcing vendor for the *SafeHome* e-commerce Web site and manager of the Web engineering team that will be doing the work.

The conversation:

Doug: Good to finally meet with you, Sharon. We've certainly got some work to do over the next month or so.

Sharon (smiling): We do, but you guys seem to have your act together. Vinod has already given us a draft specification for the site and has also defined most of the important content objects and site functionality.

Doug: Good. What else do you need?

Sharon: The e-commerce functionality is easy. The thing that worries me is the front end . . . the work required to have the user customize the product pre-purchase.

Doug: Vinod gave you the basic procedure, didn't he?

Sharon: He did, but I'd like to validate it with some real users. We'll also need to contact your content developers to get proper descriptions for each sensor, pictures, pricing, interface/interconnection info, that sort of thing.

Doug: Did Vinod have time to do a rough storyboard of the customization process for you?

Sharon: He's working on it as we speak. He had to put out a fire on the product side. He knows it's critical . . . said he'd e-mail it to me tomorrow morning.

Doug: Okay . . . look, I'd like to stay in the loop on this project. Can we establish some ground rules for changes on our end. I don't want to get in your way, but . . .

Sharon: Not a problem, we like to keep our clients involved.

Doug: I'll serve as liaison for this project. All communication will come through me or someone like Vinod that I appoint. Since we're on a tight schedule, I'd like to establish a schedule that has one-day granularity and talk or e-mail with you everyday about accomplishments, problems, etc. I know it's a lot, but that's what I think is appropriate.

Sharon: That's okay.

Doug (picking up a few pages of paper from his desktop and handing them to Sharon): I've written up a rough schedule with milestone dates . . . what do you think?

Sharon (after studying the schedule): Hmmm. I'm not sure this'll work for us. Let me work up an alternative and e-mail it to you later today.

Doug: Sure.

17.4.2 WebApp Planning—In-House Web Engineering

As WebApps become more pervasive and business strategic, many companies have opted to control development in-house. Not surprisingly, in-house WebE is managed somewhat differently that an outsourcing effort.

"What do you do when you need to have a Web site done yesterday?"
James Lewis

The management of small and moderately-sized (i.e., less than 3–5 months in duration) WebE projects requires an agile approach that deemphasizes project management but does *not* eliminate the need to plan. Basic project management principles (Chapter 21) still apply, but the overall approach is leaner and less formal. However, as the size of the WebApp project grows, Web engineering project management becomes more and more like software engineering project management (Part 4 of this book). The steps that follow are recommended for small and moderately-sized WebE projects:

ADVICE

It's important to recognize that the steps discussed in this section can be performed quickly. In no case, should WebE planning for projects of this size take more than 5 percent of overall project effort.

Understand scope, the dimensions of change, and project constraints. No project—regardless of how tight the time constraints—can begin until the project team understands what must be built. Requirements gathering and customer communication are essential precursors to effective WebApp planning.

Define an incremental project strategy. We have already noted that WebApps evolve over time. If evolution is uncontrolled and chaotic, the likelihood of a successful outcome is small. However, if the team establishes a project strategy that defines WebApp increments (releases) that provide useful content and functionality for end-users, engineering effort can be more effectively focused.

Perform risk analysis. A detailed discussion of risk analysis for traditional software engineering projects is presented in Chapter 25.[7] All risk management tasks are performed for WebE projects, but the approach to them is abbreviated.

Schedule risk and technology risk dominate the concern of most Web engineering teams. Among the many risk-related questions that the team must ask and answer are: Can planned WebApp increments be delivered within the timeframe defined? Will these increments provide on-going value for end-users while additional increments are being engineered? How will requests for change impact delivery schedules? Does the team understand the required Web engineering methods, technologies, and tools? Is the available technology appropriate for the job? Will likely changes require the introduction of new technology?

7 Those readers who are unfamiliar with basic risk management terminology and practices are urged to examine Chapter 25 at this time.

Develop a quick estimate. The focus of estimation for most Web engineering projects is on macroscopic, rather than microscopic, issues. The WebE team assesses whether planned WebApp increments can be developed with available resources according to the defined schedule constraints. This is accomplished by considering each increment's content and function as a whole. "Microscopic" functional or work breakdowns of the increment, followed by the computation of multi-data point estimates (see Chapter 23) are normally not conducted.

Select a task set (process description). Using a process framework (Chapter 16), select a set of Web engineering tasks that is appropriate for the characteristics of the problem, the product, the project, and the people on the Web engineering team. Recognize that the task set may be adapted to fit each development increment.

Establish a schedule. A WebE project schedule has relatively fine granularity for tasks to be performed in the short-term and then much more coarse granularity during later time periods (when additional increments are to be delivered). That is, Web engineering tasks are distributed along the project timeline for the increment to be developed. Task distribution for subsequent WebApp increments is delayed until delivery of the scheduled increment.

KEY POINT

Regardless of project size, it's important to establish project milestones so that progress can be assessed.

Define project tracking mechanisms. In an agile development environment, the delivery of an operational software increment is often the primary measure of progress. But long before deliverable software is available, the Web engineer will inevitably encounter the question, Where are we? In conventional software engineering work, progress is measured by determining which milestones (e.g., a successful review of a work product) have been achieved. For small and moderately-sized Web engineering projects, milestones may be less well-defined, and formal quality assurance activities may be de-emphasized. Therefore, an answer can be derived by polling the Web engineering team to determine which framework activities have been completed. However, this approach can be unreliable. Another approach is to determine how many use-cases have been implemented and how many use-cases (for a given increment) remain to be implemented. This provides a rough indication of the relative degree of "completeness" of the project increment.

"Progress is made by correcting the mistakes resulting from the making of progress."

Claude Gibb

Establish a change management approach. Change management is facilitated by the incremental development strategy that has been recommended for WebApps. Because the development time for an increment is short, it is often possible to delay the introduction of a change until the next increment, thereby reducing the delaying effects associated with changes that must be implemented "on the fly." A discussion of configuration and content management for WebApps is presented in Chapter 27.

SOFTWARE TOOLS

WebE Project Management

Objective: To assist a Web engineering team in planning, managing, controlling, and tracking Web engineering projects.

Mechanics: Project management tools enable a WebE team to establish a set of work tasks, assign effort and specific responsibility for each task, establish task dependencies, define a schedule, and track and control project activities. Many tools in this category are Web-based.

Representative Tools[8]

Business Engine, developed by Business Engine (www.businessengine.com), is a suite of Web-based tools that provide full project management facilities for WebE and conventional software projects.

Iteamwork, developed by iTeamwork.com (www.iteamwork.com), "is a free, on-line, Web-based team project management application that you use with your web browser."

OurProject, developed by Our Project (www.ourproject.com), is a suite of project management tools that are applicable to WebE and conventional software projects.

Proj-Net, developed by Rational Concepts, Inc. (www.rationalconcepts.com), implements a "virtual project office (VPO) for collaboration and communication."

StartWright (www.startwright.com/project1.htm) has developed one of the Web's most comprehensive resources for both WebE and conventional software project management tools and information.

It should be noted that many conventional project management tools (Part 4 of this book) can also be used effectively for WebE projects.

17.5 Metrics for Web Engineering and WebApps

Web engineers develop complex systems, and like other technologists who perform this task, they should use metrics to improve the Web engineering process and product. In Chapter 15, we discussed the strategic and tactical uses for software metrics in a software engineering context. These uses also apply to Web engineering.

To summarize, software metrics provide a basis for improving the software process, increasing the accuracy of project estimates, enhancing project tracking, and improving software quality. Web engineering metrics could, if properly characterized, achieve all these benefits and also improve usability, WebApp performance, and user satisfaction.

ADVICE

In general, the number of WebE metrics that you should collect and their overall complexity should be directly proportional to the size of the WebApp that is to be built.

In the context of Web engineering, metrics have three primary goals: (1) to provide an indication of the quality of the WebApp from a technical point of view, (2) to provide a basis for effort estimation, and (3) to provide an indication of the success of the WebApp from a business point of view.

In this section, we summarize a set of common effort and complexity metrics[9] for WebApps. These may be used to develop a historical database for effort estimation. In addition, complexity metrics may ultimately lead to an ability to quantitatively assess one or more of the technical attributes of WebApps discussed in Chapter 16.

8 Tools noted here do not represent an endorsement, but rather a sampling of tools in this category. In most cases, tool names are trademarked by their respective developers.

9 It is important to note that WebE metrics are still in their infancy.

17.5.1 Metrics for Web Engineering Effort

Web engineers expend human effort performing a variety of work tasks as a WebApp evolves. Mendes and her colleagues [MEN01] suggest a number of possible effort measures for WebApps. Some or all of these could be recorded by a Web engineering team and later used to build a historical database for estimation (Chapter 23).

Application Authoring and Design Tasks

Suggested measure	Description
structuring effort	time to structure WebApp and/or derive architecture
interlinking effort	time to interlink pages to build the WebApp's structure
interface planning	time taken to plan WebApp's interface
interface building	time taken to implement WebApp's interface
link-testing effort	time taken to test all links in WebApp
media-testing effort	time taken to test all media in WebApp
total effort	structuring effort + interlinking effort + interface planning + interface building + link-testing effort + media-testing effort

Page Authoring

Suggested measure	Description
text effort	time taken to author or reuse text in page
page-linking effort	time taken to author links in page
page-structuring effort	time taken to structure page
total page effort	text effort + page-linking effort + page-structuring effort

Media Authoring

Suggested measure	Description
media effort	time taken to author or reuse media files
media-digitizing effort	time taken to digitize media
total media effort	media effort + media-digitizing effort

Program Authoring

Suggested measure	Description
programming effort	time taken to author HTML, Java, or related language implementations
reuse effort	time to reuse/modify existing programming

17.5.2 Metrics for Assessing Business Value

WebRef
An excellent reference on many subjects related to Internet business is **www.internet.com.**

It's interesting to note that business people have considerably outpaced Web engineers in developing, collecting, and using metrics for WebApps (e.g. [STE02], [NOB01]). By understanding the demographics of end-users and their usage patterns, a company or organization can develop immediate input for more meaningful WebApp content, more effective sales and marketing efforts, and better profitability for the business.

The mechanisms required to collect business value data are often implemented by the Web engineering team, but evaluation of the data and actions that result are performed by other constituencies. For example, assume that the number of page views can be determined for each unique visitor. Based on metrics collected, visitors arriving from search engine *X* average nine page views while visitors from portal *Y* have only two page views. These averages can be used by the marketing department to allocate banner advertising budgets (advertising at search engine *X* provides greater exposure, based on metrics collected, than advertising at portal *Y*).

A complete discussion of the collection and use of business value metrics (including the on-going debate about personal privacy) is beyond the scope of this book. The interested reader should examine [INA02], [EIS02], [PAT02] or [RIG01].

SOFTWARE TOOLS

Web Metrics

Objective: To assess the manner in which a WebApp is being used, the categories of users, and the usability of the WebApp.

Mechanics: The vast majority of Web metrics tools capture usage information once the WebApp goes on-line. These tools provide a broad array of data that can be used to assess which elements of the WebApp are most used, how they are used, and who uses them.

Representative Tools[10]

Clicktracks, developed by clicktracks.com (www.clicktracks.com), is a log file analysis tool that displays Web site visitor behavior directly on pages of the Web site.

Marketforce, developed by Coremetrics (www.Coremetrics.com), is representative of many tools that collect data that can be used to assess the success of e-commerce WebApps.

Web Metrics Testbed, developed by NIST (zing.ncsl.nist.gov/WebTools/), is a suite of Web-based tools that assess the usability of a WebApp.

WebTrends, developed by netIQ (www.NetIQ.com), collects a broad range of usage data for WebApps of all types.

17.6 "Worst Practices" for WebApp Projects

Sometimes the best way to learn how to do something correctly is to examine how not to do it! Over the past decade, more than a few WebApps have failed because (1) a disregard for project and change management principles (however informal) resulted

10 Tools noted here do not represent an endorsement, but rather a sampling of tools in this category. In most cases, tool names are trademarked by their respective developers.

in a Web engineering team that "bounced off the walls"; (2) an ad hoc approach to WebApp development failed to yield a workable system; (3) a cavalier approach to requirements gathering and analysis failed to yield a system that met user needs; (4) an incompetent approach to design failed to yield development of a WebApp that was usable, functional, extensible (maintainable), and testable; (5) an unfocused approach to testing failed to yield a system that worked prior to its introduction.

With these realities in mind, it might be worthwhile to consider a set of Web engineering "worst practices," adapted from an article by Tom Bragg [BRA00]. If your e-project exhibits any of them, immediate remedial action is necessary.

Worst practice #1: *We have a great idea, so let's begin building the WebApp—now.* Don't bother considering whether the WebApp is business justified, whether users will really want to use it, whether you understand the business requirements. Time is short, we have to start.

Reality: Take a few hours/days and make a business case for the WebApp. Be sure that the idea is endorsed by those who will fund it and those who will use it.

Worst practice #2: *Stuff will change constantly, so there's no point in trying to understand WebApp requirements.* Never write anything down (wastes time). Rely solely on word of mouth.

Reality: It is true that WebApp requirements evolve as Web engineering activities proceed. It's also fast and simple to convey information verbally. However, a cavalier approach to requirements gathering and analysis is a catalyst for even more (unnecessary) change.

Worst practice #3: *Developers whose dominant experience has been in traditional software development can develop WebApps immediately. No new training is required.* After all, software is software, isn't it?

Reality: WebApps are different. A broad array of methods, technologies, and tools must be expertly applied. Training and experience with them is essential.[11]

Worst practice #4: *Be bureaucratic.* Insist on leaden process models, time sheets, lots of unnecessary "progress" meetings, and project leaders who have never managed a WebApp project.

Reality: Encourage an agile process that emphasizes the competence and creativity of an experienced Web engineering team. Then get out of the way and let them do the work. If project-related data must be collected (for legal reasons or for the computation of metrics), data entry/collection should be as simple and unobtrusive as possible.

Worst practice #5: *Testing? Why bother?* We'll give it to a few end-users and let them tell us what works and what doesn't.

11 Many large WebE projects require integration with conventional applications and databases. In such instances, individuals with only conventional experience can and should be involved.

Reality: Over time, end-users do perform thorough "tests," but they are so upset by unreliability and poor performance that they leave (never to return) long before problems are corrected.

In the chapters that follow, we consider Web engineering methods that will help you avoid these mistakes.

17.7 Summary

Formulation is a customer communication activity that defines the problem that a WebApp is to solve. Business need, project goals and objectives, end-user categories, major functions and features, and the degree of interoperability with other applications are all identified. As more detailed and technical information is acquired, formulation becomes requirements analysis.

The WebE team is composed of a group of technical and nontechnical members who are organized in a manner that gives them considerable autonomy and flexibility. Project management is required during Web engineering, but project management tasks are abbreviated and considerably less formal than those applied for conventional software engineering projects. Many WebApp projects are outsourced, but there is a growing trend toward in-house WebApp development. Project management for each approach differs in both strategy and tactics.

Web engineering metrics are in their infancy but have the potential to provide an indication of the WebApp quality, provide a basis for effort estimation, and provide an indication of the success of the WebApp from a business point of view.

References

[BRA00] Bragg, T., "Worst Practices for e-Business Projects: We Have Met the Enemy and He Is Us!" *Cutter IT Journal,* vol. 13, no. 4, April 2000, pp. 35–39.

[CON02] Constantine, L., and L. Lockwood, "User-Centered Engineering for Web Applications," *IEEE Software,* vol. 19, no. 2, March/April 2002, pp. 42–50.

[EIS02] Eisenberg, B., "How to Interpret Web Metrics," *ClickZ Today,* March 2002, available at http://www.clickz.com/sales/traffic/article.php/992351.

[FUC02a] Fuccella, J., J. Pizzolato, and J. Franks, "Finding Out What Users Want from your Web Site," IBM developerWorks, 2002, http://www-106.ibm.com/developerworks/library/moderator-guide/requirements.html.

[FUC02b] Fuccella, J., and J. Pizzolato, "Giving People What They Want: How to Involve Users in Site Design," IBM developerWorks, 2002, http://www-106.ibm.com/developerworks/library/design-by-feedback/expectations.html.

[GNA99] Gnado, C., and F. Larcher, "A User-Centered Methodology for Complex and Customizable Web Applications Engineering," *Proc. First ICSE Workshop in Web Engineering,* ACM, Los Angeles, May 1999.

[HAN99] Hansen, S., Y. Deshpande, and S. Murugesan, "A Skills Hierarchy for Web Information System Development," *Proc. First ICSE Workshop on Web Engineering,* ACM, Los Angeles, May 1999.

[INA02] Inan, H., and M. Kean, *Measuring the Success of Your Web Site,* Longman Publishing, 2002.

[KID00] Kidder, T., *The Soul of a New Machine,* Back Bay Books (reprint edition), 2000.

[KUL00] Kulik, P., and R. Samuelsen, "e-Project Management for a New e-Reality," Project Management Institute, December, 2000, http://www.seeprojects.com/e-Projects/ e-projects.html.

[LOW98] Lowe, D., and W. Hall, eds., *Hypertext and the Web—An Engineering Approach,* Wiley, 1998.

[MEN01] Mendes, E., N. Mosley, and S. Counsell, "Estimating Design and Authoring Effort," *IEEE Multimedia,* January-March 2001, pp. 50-57.

[NOB01] Nobles, R., and K. Grady, *Web Site Analysis and Reporting,* Premier Press, 2001.

[PAT02] Patton, S., "Web Metrics That Matter," *CIO,* November 15, 2002. available at http://www.computerworld.com/developmenttopics/websitemgmt/story/0,10801,76002,00.html.

[PIC01] Pickering, C., "Building an Effective E-Project Team," *E-Project Management Advisory Service,* Cutter Consortium, vol. 2, no. 1, 2001, http://www.cutter.com/ consortium.

[POW98] Powell, T.A., *Web Site Engineering,* Prentice-Hall, 1998.

[RIG01] Riggins, F., and S. Mitra, "A Framework for Developing E-Business Metrics through Functionality Interaction, January 2001, download from http:// digitalenterprise.org/metrics/metrics.html.

[STE02] Sterne, J., *Web Metrics: Proven Methods for Measuring Web Site Success,* Wiley, 2002.

[TIL99] Tilley, S., and S. Huang, "On the Emergence of the Renaissance Software Engineer," *Proc. 1st ICSE Workshop on Web Engineering,* ACM, Los Angeles, May 1999.

Problems and Points to Ponder

17.1. Consider the metrics for Web engineering effort discussed in Section 17.5.1. Try to develop five or more additional metrics for one or more categories.

17.2. Three fundamental formulation questions are posed in Section 17.1.1. Are there any other questions that you think might be asked at this point? If so what are they, and why would you ask them?

17.3. The ease of navigation through a Web site is an important indicator of WebApp quality. Develop two or three metrics that could be used to indicate the ease of navigation.

17.4. In your own words, discuss how information gathered during customer communication is "analyzed" and what the output from this activity is.

17.5. Describe five risks associated with outsourcing WebApp development.

17.6. Considering the *SafeHome* e-commerce site discussed in this chapter, what user communication mechanism would you use to elicit system requirements, and why?

17.7. Review the table presented in Figure 17.1. Add three more rows that will further distinguish traditional projects from e-projects.

17.8. Describe the role of the Web publisher in your own words.

17.9. Review the characteristics of agile development teams discussed in Chapter 4. Do you feel that an agile team organization is appropriate for WebE? Would you make any changes to the organization for WebApp development?

17.10. What benefits can be derived from requiring the development of use-cases as part of the requirements gathering activity?

17.11. Describe five risks associated with in-house WebApp development.

17.12. In the context of requirements gathering, what is a "user category"? Give examples of three user categories for an on-line book seller.

17.13. How does formulation differ from requirements gathering? How does formulation differ from requirements analysis and analysis modeling?

17.14. Using one of the references noted in Section 17.5.2, discuss how business value metrics can be used to assist in pragmatic business decision making.

Further Readings and Information Sources

Methods for WebApp formulation and requirements gathering can be adapted from discussions of similar methods for conventional application software. Further readings recommended in Chapters 7 and 8 contain much useful information for the Web engineer.

Flor (*Web Business Engineering,* Addison-Wesley, 2000) discusses business analysis and related concerns that enable the Web engineer to better understand customer needs. WebApp usability is a concept that underlies much of the information defined as part of formulation and requirements gathering. Krug and Black (*Don't Make Me Think: A Common Sense Approach to Web Usability,* Que Publishing, 2000) contains many guidelines and examples that can help the Web engineer translate user requirements into an effective WebApp.

Project management for WebE projects draws from many of the same principles and concepts that are applied for conventional software projects. However, agility is a watchword. Wallace (*Extreme Programming for Web Projects,* Addison-Wesley, 2003) describes how agile development can be used for WebE and contains useful discussions of project management issues. Shelford and Remillard (*Real Web Project Management,* Addison-Wesley, 2003), O'Connell (*How to Run Successful Projects in Web Time,* Artech House, 2000), Freidlein (*Web Project Management,* Morgan Kaufman, 2000), and Gilbert (*90 Days to Launch: Internet Projects on Time and on Budget,* Wiley, 2000) discuss a wide array of project management issues for WebE. Whitehead (*Leading a Software Development Team,* Addison-Wesley, 2001) presents many useful guidelines that can be adapted for Web engineering teams.

Techniques for using Web metrics in business decision making are presented in books by Sterne [STE02], Inan [INA02], Nobles [NOB01] and Menasce and Almeida (*Capacity Planning for Web Services: Metrics, Models and Methods,* Prentice-Hall, 2001). "Worst practices" are considered by Ferry and Ferry (*77 Sure-Fire Ways to Kill a Software Project,* iUniverse.com, 2000).

A wide variety of information sources on formulation and planning for Web engineering is available on the Internet. An up-to-date list of World Wide Web references that are relevant to formulation and planning can be found at the SEPA Web site:
http://www.mhhe.com/pressman.

CHAPTER 18

ANALYSIS FOR WEBAPPS

KEY CONCEPTS
analysis model
configuration model
content model
content relationships
data tree
functional model
interaction model
navigation analysis
relationship analysis
RNA
use-cases
user hierarchy

At first glance, there is an apparent contradiction when we consider analysis modeling within the context of Web engineering. After all, we have noted (Chapter 16) that WebApps have an immediacy and a volatility that mitigate against detailed modeling at either the analysis or the design level. And if we do any modeling at all, the agile philosophy (an appropriate process model for many Web engineering projects) suggests that analysis modeling is downplayed in favor of limited design modeling. Franklin [FRA02] notes this situation when he writes:

> Web sites are typically complex and highly dynamic. They require short development phases in order to get the product up and running quickly. Frequently, developers go straight to the coding phase without really understanding what they are trying to build or how they want to build it. Server-side coding is often done *ad hoc,* database tables are added as needed, and the architecture evolves in a sometimes unintentional manner. But some modeling and disciplined software engineering can make the software development process much smoother and ensure that the Web system is more maintainable in the future.

Is it possible to have it both ways? Can we do "some modeling and disciplined software engineering" and still work effectively in a world where immediacy and volatility reign? The answer is a qualified yes.

QUICK LOOK

What is it? The analysis of a potential Web application focuses on three important questions: (1) what information or content is to be presented or manipulated; (2) what functions are to be performed for the end-user; and (3) what behaviors will the WebApp exhibit as it presents content and performs functions? The answers to these questions are represented as part of an analysis model that encompasses a variety of UML representations.

Who does it? Web engineers, nontechnical content developers, and stakeholders participate in the creation of the analysis model.

Why is it important? Throughout this book we have emphasized the need to understand the problem before you begin to solve it. Analysis modeling is important not because it enables a Web engineering team to develop a concrete model of WebApp requirements (things change too frequently for this to be a realistic expectation), but rather, analysis modeling enables a Web engineer to define fundamental aspects of the problem—things that are unlikely to change (in the near term). When fundamental content, function, and behavior are understood, design and construction are facilitated.

What are the steps? Analysis modeling focuses on four fundamental aspects of the problem—content, interaction, function, and configuration. Content analysis identifies content classes and collaborations. Interaction analysis describes basic elements of user interaction, navigation, and the system behaviors that occur as a consequence. Function analysis defines the WebApp functions that are performed for the user and the sequence of processing that occurs as a consequence. Configuration analysis identifies the operational environment(s) in which the WebApp resides.

What is the work product? The analysis model is comprised of a set of UML diagrams and text that describe content, interaction, function, and configuration.

How do I ensure that I've done it right? Analysis modeling work products must be reviewed for correctness, completeness, and consistency.

A Web engineering team should embrace analysis modeling when most or all of the following conditions are met:

- The WebApp to be built is large and/or complex.
- The number of stakeholders is large.
- The number of Web engineers and other contributors is large.
- The goals and objectives (determined during formulation) for the WebApp will effect the business' bottom line.
- The success of the WebApp will have a strong bearing on the success of the business.

If these conditions are not present, it is possible to de-emphasize analysis modeling, using information obtained during formulation and requirements gathering (Chapter 17) as the basis for creating a design model for the WebApp. In such circumstances, limited analysis modeling may occur, but it will be rolled into design.

18.1 Requirements Analysis for WebApps

Requirements analysis for WebApps encompasses three major tasks: formulation, requirements gathering,[1] and analysis modeling. During formulation, the basic motivation (goals) and objectives for the WebApp are identified, and the categories of users are defined. As requirements gathering begins, communication between the Web engineering team and WebApp stakeholders (e.g., customers, end-users) intensifies. Content and functional requirements are listed and interaction scenarios (use-cases) written from the end-user's point-of-view are developed. The intent is to establish a basic understanding of why the WebApp is to be built, who will use it, and what problem(s) it will solve for its users.

"The engineering principles of planning before designing and designing before building have withstood every prior technology transition; they'll survive this transition as well."

Watts Humphrey

1 Formulation and requirements gathering are discussed in detail in Chapter 17.

18.1.1 The User Hierarchy

The categories of end-users who will interact with the WebApp are identified as part of the formulation and requirements gathering tasks. In most cases, user categories are relatively limited and a UML representation of them is unnecessary. However, when the number of user categories grows, it is sometimes advisable to develop a *user hierarchy* as shown in Figure 18.1. The figure depicts users for the SafeHome-Assured.com e-commerce site discussed in Chapters 16 and 17.

ADVICE

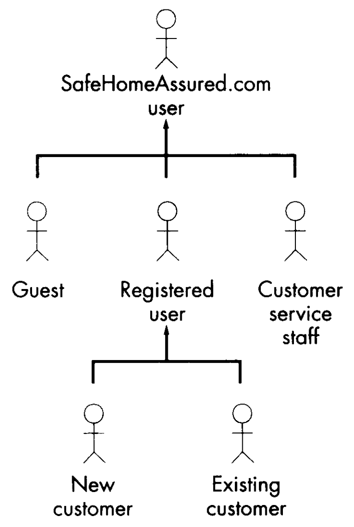
It's a good idea to build a user hierarchy. It provides you with a snapshot of the user population and a cross-check to help ensure that the needs of every user have been addressed.

The user categories (often called *actors*) shown in Figure 18.1 provide an indication of the functionality to be provided by the WebApp and indicate a need for use-cases to be developed for each end-user (actor) noted in the hierarchy. Referring to the figure, the **SafeHomeAssured.com user** at the top of the hierarchy represents the most general user class (category) and is refined in levels below. A **guest** is a user who visits the site but does not register. Such users are often searching for general information, comparison shopping, or otherwise interested in "free" content or functionality. A **registered user** takes the time to provide contact information (along with other demographic data requested by forms input). Subcategories for **registered user** include:

- **new customer**—a registered user who wants to customize and then purchase *SafeHome* components (and hence, must interact with the WebApp e-commerce functionality);
- **existing customer**—a user who already owns *SafeHome* components and is using the WebApp to (1) purchase additional components; (2) to acquire tech support information; or (3) contact customer support.

FIGURE 18.1

User hierarchy for SafeHome-Assured.com



Members of **customer service staff** are special users who can also interact with SafeHomeAssured.com content and functionality as they assist customers who have contacted *SafeHome* customer support.

18.1.2 Developing Use-Cases

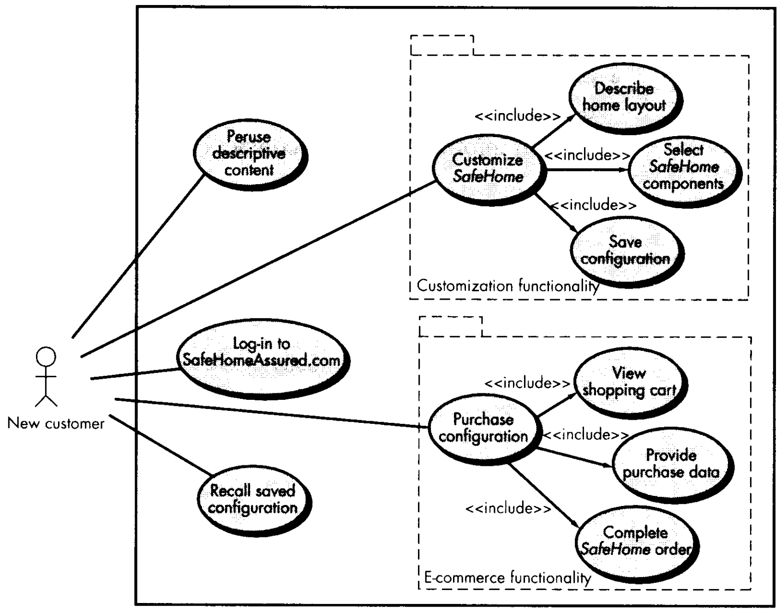
Franklin [FRA01] refers to use-cases as "bundles of functionality." This description captures the essence of this important analysis modeling technique.[2] Use-cases are developed for each user category described in the user hierarchy. In the context of Web engineering, the use-case itself is relatively informal—a narrative paragraph that describes a specific interaction between a user and the WebApp.[3]

Figure 18.2 represents a UML use-case diagram for the **new customer** user category (Figure 18.1). Each oval in the diagram represents a use-case that describes a specific interaction between **new customer** and the WebApp. For example, the first interaction is described by the *log-in to SafeHomeAssured.com* use-case. No more than a single paragraph would be required to describe this common interaction.

Major WebApp functionality (and the use-cases that are relevant for it) are noted inside the dashed boxes in Figure 18.2. These are referred to a "packages" in

FIGURE 18.2

Use-case diagram for new-customer



2 Techniques for developing use-cases have been discussed in detail earlier in this book (see Chapters 7 and 8).

3 Although it is possible to develop more formal use-case descriptions, the need for WebE agility often precludes this approach.

UML and represent specific functionality. Two packages are noted: *customization* and *e-commerce*.

As an example, we consider the *customization* package of use-cases. A new customer must describe the home environment into which *SafeHome* will be installed. To accomplish this, the use-cases *describe home layout, select SafeHome hardware,* and *save configuration* are initiated by **new customer.** Consider these preliminary use-cases written from the point of view of a **new customer:**

ADVICE

As the size of a WebApp grows and analysis modeling becomes more rigorous, the preliminary use-cases presented here would have to be expanded to conform more closely to the format suggested in Section 8.5 of Chapter 8.

Use-case: *describe home layout*

The WebApp will ask some general questions about the environment in which I plan to install *SafeHome*—the number of rooms, size of rooms, type of room, the number of floors, the number of exterior doors and windows. The WebApp will enable me to build a rough floor plan by putting together outline shapes of the rooms for each floor. I'll be able to give the floor plan a name and save it for future reference (see use-case: *save configuration*).

Use-case: *select SafeHome components*

The WebApp will then recommend product components (e.g., control panels, sensors, cameras) and other features (e.g., PC-based functionality implemented in software) for each room and exterior entrance. If I request alternatives, the WebApp will provide them, if they exist. I will be able to get descriptive and pricing information for each product component. The WebApp will create and display a bill-of-materials as I select various components. I'll be able to give the bill-of-materials a name and save it for future reference (see use-case: *save configuration*).

Use-case: *save configuration*

The WebApp will allow me to save customization data so that I can return to it later. I can save the house layout and the *SafeHome* bill-of materials I choose for the layout. To accomplish this, I provide a unique identifier for the house layout and the bill-of-materials. I also provide a special configuration password that must be validated before I can gain access to the saved information.

Although considerably more detail could be provided for each of these use-cases, the informal text description provides useful insight. Similar descriptions would be developed for each oval in Figure 18.2.

SAFEHOME

Refining Use-Cases for WebApps

The scene: Doug Miller's office.

The players: Doug Miller (manager of the *SafeHome* software engineering group), Sharon Woods, manager of the outsourcing vendor's Web engineering team for the *SafeHome* e-commerce Web site, and Sam Chen, manager of the SafeHomeAssured.com customer support organization.

The conversation:

Doug: Glad to hear things are progressing well, Sharon. Analysis modeling is almost complete?

Sharon (smiling): We're making progress. The only set of use-cases left to develop from the user hierarchy (Figure 18.1) is the *customer service staff* category.

Doug (looking at Sam): And you have those now, Sam?

Sam: I do. I've e-mailed them to you, Sharon, and cc'd you, Doug. Here's the hardcopy version. (He hands sheets of paper to Doug and Sharon.)

Sam: The way we look at it, we want to use the SafeHomeAssured.com Web site as a support tool when customers phone in an order. Our phone reps will complete all necessary forms, etc. and process the order for the customer.

Doug: Why not just refer the customer to the Web site?

Sam (smiling): You techies think that everyone is comfortable with the Web. They're not! Plenty of people still like the telephone, so we have to give them that option. But we don't want to build a separate order processing system when most of the pieces are already in place on the Web.

Sharon: Makes sense.

(All parties read the use-cases [an example follows]):

Use-case: *describe home layout* [note that this differs from the use-case of the same name for *new customer* category]

I will ask the customer (via the phone) to describe each room of the house and will enter room dimensions and other characteristics on one big form designed specifically for customer support personnel. Once the house data are entered I can save the data under the customer's name or phone number.

Sharon: Sam, you've been kind of terse in your preliminary use-case desriptions. I think we're going to need to flesh them out a bit.

Doug (nodding): I agree.

Sam (frowning): How so?

Sharon: Well . . . you mention "one big form designed specifically for customer support personnel." We're going to need more detail.

Sam: What I meant was that we don't need to walk our reps through the process like you do for an online customer. One big form should do the trick.

Sharon: Let's sketch out what the form should look like.

The parties work to provide sufficient detail to allow Sharon's team to make effective use of the use-cases.

18.1.3 Refining the Use-Case Model

As use-case diagrams are created for each user category, a top-level view of externally observable WebApp requirements is developed. Use-cases are organized into functional packages, and each package is assessed [CON00] to ensure that it is:

How do we assess packages of use-cases that have been grouped by user function?

- *Comprehensible*—all stakeholders understand the purpose of the package.
- *Cohesive*—the package addresses functions that are closely related to one another.
- *Loosely coupled*—functions or classes within the package collaborate with one another, but collaboration outside the package is kept to a minimum.
- *Hierarchically shallow*—deep functional hierarchies are difficult to navigate and hard for end-users to understand; therefore, the number of levels within a use-case hierarchy should be minimized whenever possible.

Because requirements analysis and modeling are iterative activities, it is likely that new use-cases will be added to packages that have been defined, that existing use-cases will be refined, and that specific use-cases might be reallocated to different packages.

18.2 The Analysis Model for WebApps

A WebApp *analysis model* is driven by information contained within the use-cases that have been developed for the application. Use-case descriptions are parsed to identify potential analysis classes and the operations and attributes associated with each class. Content to be presented by the WebApp is identified, and functions to be performed are extracted from the use-case descriptions. Finally, implementation-specific requirements should be developed so that the environment and infrastructure that support the WebApp can be built.

Four analysis activities—each contributing to the creation of a complete analysis model are:

? What types of analysis activity occur during modeling of a WebApp?

- *Content analysis* identifies the full spectrum of content to be provided by the WebApp. Content includes text, graphics and images, and video and audio data.
- *Interaction analysis* describes the manner in which the user interacts with the WebApp.
- *Functional analysis* defines the operations that will be applied to WebApp content and describes other processing functions that are independent of content but necessary to the end-user.
- *Configuration analysis* describes the environment and infrastructure in which the WebApp resides.

The information collected during these four analysis tasks should be reviewed, modified as required, and then organized into a model that can be passed to WebApp designers.

The model itself contains structural and dynamic elements. *Structural elements* identify the analysis classes and content objects that are required to create a WebApp that meets stakeholders needs. The *dynamic elements* of the analysis model describe how the structural elements interact with one another and with end-users.

"Successful [WebApps] allow customers to meet their needs better, faster, or cheaper themselves, rather than working through [a company's] employee end-users."

Mark McDonald

18.3 The Content Model

The *content model* contains structural elements that provide an important view of content requirements for a WebApp. These structural elements encompass content objects (e.g., text, graphical images, photographs, video images, audio) that are presented as part of the WebApp. In addition, the content model includes all analysis classes—user-visible entities that are created or manipulated as a user interacts with the WebApp. An analysis class encompasses attributes that describe it, operations that effect behavior required of the class, and collaborations that allow the class to communicate with other classes.

Like other elements of the analysis model, the content model is derived from a careful examination of use-cases developed for the WebApp. Use-cases are parsed to extract content objects and analysis classes.

18.3.1 Defining Content Objects

KEY POINT
A content object is any item of cohesive information that is to be presented to an end-user. Typically, content objects are extracted from use-cases.

Web applications present pre-existing information—called *content*—to an end-user. The type and form of content spans a broad spectrum of sophistication and complexity. Content may be developed prior to the implementation of the WebApp, while the WebApp is being built, or long after the WebApp is operational. In every case, it is incorporated via navigational reference into the overall WebApp structure. A *content object* might be a textual description of a product, an article describing a news event, an action photograph taken at a sporting event, an animated representation of a corporate logo, a short video of a speech, or an audio overlay for a collection of Powerpoint slides.

Content objects are extracted from use-cases by examining the scenario description for direct and indirect references to content. For example, in the use-case *select SafeHome components,* we encounter the sentence:

I will be able to get descriptive and pricing information for each product component.

Although there is no direct reference to content, it is implied. The Web engineer would meet with the author of the use-case and gain a more detailed understanding of what "descriptive and pricing information" means. In this case, the author of the use-case might indicate that "descriptive information" includes (1) a one paragraph general description of the component; (2) a photograph of the component; (3) a multiparagraph technical description of the component; (4) a schematic diagram of the component showing how it fits into a typical *SafeHome* system, and (5) a thumbnail video that shows how to install the component in a typical household setting.

It is important to note that each of these content objects must be developed (often by content developers who are *not* Web engineers) or acquired for integration into the WebApp architecture (discussed in Chapter 19).

"The Web—so much content, so little time."

Author unknown

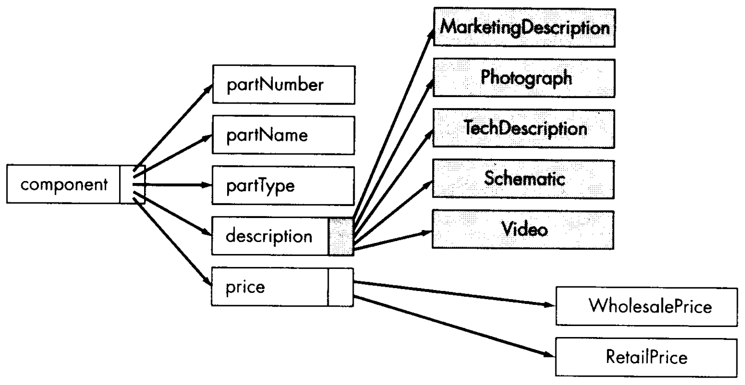
18.3.2 Content Relationships and Hierarchy

KEY POINT
A *data tree* represents a hierarchy of content objects.

In many instances, a simple list of content objects, coupled with a brief description of each object, is sufficient to define the requirements for content that must be designed and implemented. However, in some cases, the content model may contain

FIGURE 18.3

Data tree for a *SafeHome* component



entity relationship diagrams (Chapter 8) or *data trees* [SRI01] that depict the relationships among content objects and/or the hierarchy of content maintained by a WebApp.

Consider the data tree created for a *SafeHome* component shown in Figure 18.3. The tree represents a hierarchy of information that is used to describe the component (later we will see that a *SafeHome* component is actually an analysis class for this application). Simple or composite data items (one or more data values) are represented as unshaded rectangles. Content objects are represented as shaded rectangles. In the figure, **description** is defined by five content objects (the shaded rectangles). In some cases, one or more of these objects would be further refined as the data tree expands.

18.3.3 Analysis Classes[4] for WebApps

As we have already noted, analysis classes are derived by examining each use-case. For example, consider the preliminary use-case: *select SafeHome components* presented in Section 18.1.2.

Use-case: *select SafeHome components*

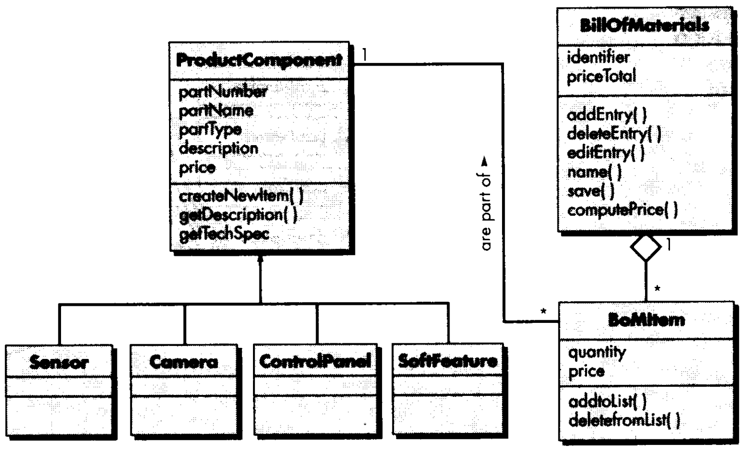
The WebApp will then recommend product components (e.g., control panels, sensors, cameras) and other features (e.g., PC-based functionality implemented in software) for each room and exterior entrance. If I request alternatives, the WebApp will provide them, if they exist. I will be able to get descriptive and pricing information for each product component. The WebApp will create and display a bill-of-materials as I select various components. I'll be able to give the bill-of-materials a name and save it for future reference (see use-case: *save configuration*).

A quick grammatical parse of the use-case identifies two candidate classes (underlined): **ProductComponent** and **BillOfMaterials.** A first-cut description of each class is shown in Figure 18.4.

4 A detailed discussion of the mechanics for identifying and representing analysis classes has been presented in Chapter 8. If you have not already done so, review Chapter 8 at this time.

FIGURE 18.4

Analysis classes for use-case: *select SafeHome components*



The **ProductComponent** class encompasses all *SafeHome* components that may be purchased to customize the product for a particular installation. It is a generalized representation of **Sensor, Camera, ControlPanel,** and **SoftFeature.** Each **ProductComponent** object contains information corresponding to the data tree shown in Figure 18.3 for the class. Some of these class attributes are single or composite data items and others are content objects (see Figure 18.3). Operations relevant to the class are also shown.

The **BillOfMaterials** class encompasses a list of components that **new customer** has selected. **BillOfMaterials** is actually an aggregation of **BoMItem** (many instances of **BoMItem** comprise one **BillOfMaterials**)—a class that builds a list composed of each component to be purchased and specific attributes about the component as shown in Figure 18.4.

Each use-case identified for SafeHomeAssured.com is parsed for analysis objects. Class models similar to the one described in this section are developed for each use-case.

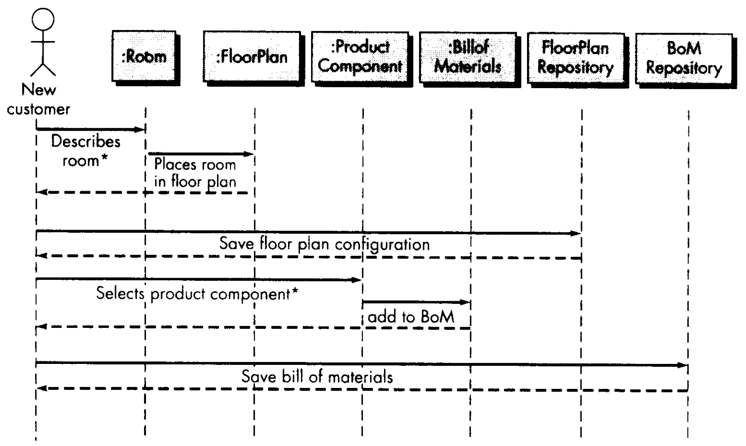
18.4 THE INTERACTION MODEL

The vast majority of WebApps enable a "conversation" between an end-user and application functionality, content, and behavior. This *interaction model* is composed of four elements: (1) use-cases, (2) sequence diagrams, (3) state diagrams,[5] and (4) a user interface prototype. In addition to these representations, the interaction is also represented within the context of the navigation model (Section 18.7).

5 Each of these is an important UML notation and has been described in Chapter 8.

FIGURE 18.5

Sequence diagram for use-case: *select SafeHome components*



ADVICE

The techniques associated with task analysis (Chapter 12) can be used to help define modes of user interaction.

Use-cases. Use-cases are the dominant element of the interaction model for WebApps. It is not uncommon to describe 100 or more use-cases when large, complex WebApps are analyzed, designed, and constructed. However, a relatively small percentage of these use-cases describe the major interactions between end-user categories (actors) and the system. Other use-cases refine the interactions, providing the analysis detail necessary to guide design and construction.

Sequence diagrams. UML sequence diagrams provide a shorthand representation of the manner in which user actions (the dynamic elements of a system defined by use-cases) collaborate with analysis classes (the structural elements of a system defined by class diagrams). Since analysis classes are extracted from use-case descriptions, there is a need to ensure that traceability exists between the classes that have been defined and the use-cases that describe system interaction.

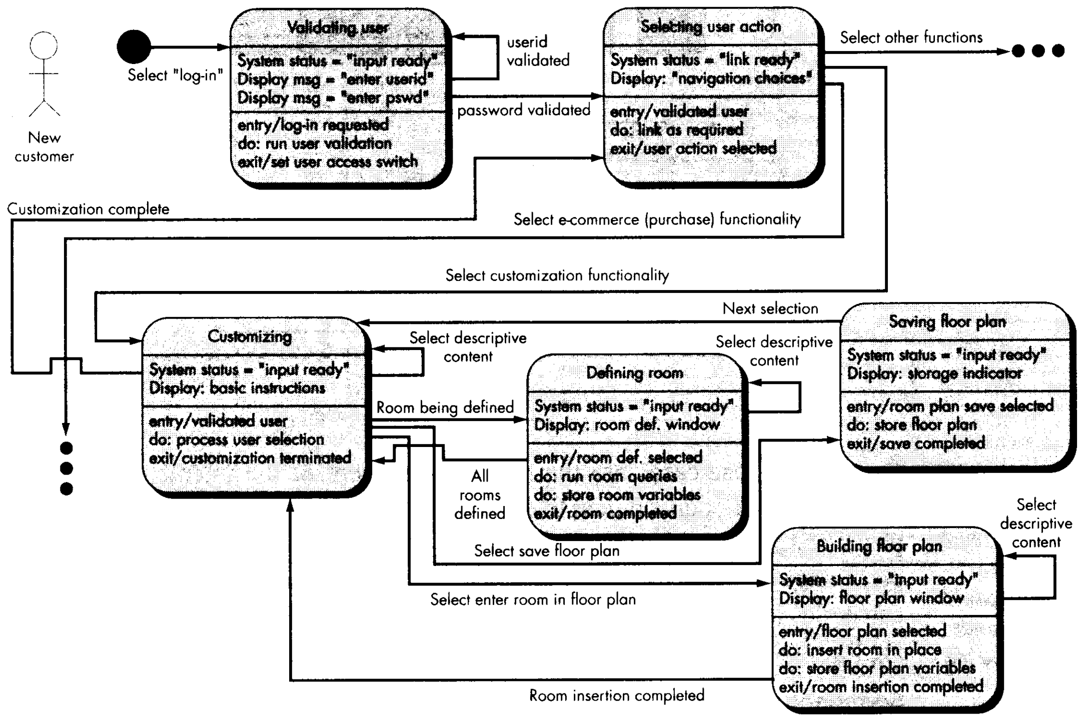
In earlier chapters we noted that sequence diagrams provide a link between the actions described in the use-case and the analysis classes (structural entities). Conallen [CON00] notes this when he writes: "The merging of dynamic and structural elements of the [analysis] model is the key link in the traceability of the model and should be taken very seriously."

ADVICE

In some cases, excerpts from the actual text of the use-case can be reproduced in the left hand column (below the user) so that direct traceability can be represented.

A sequence diagram for the *select SafeHome components* use-case is shown in Figure 18.5. The vertical axis of the diagram depicts actions that are defined within the use-case. The horizontal axis identifies the analysis classes that are used as the use-case proceeds. For example, a new customer must first describe each room within the house (the asterisk following "describe room" indicates that the action is iterative). To accomplish this, the new customer answers questions about the room's size, doors, and windows, and so forth. Once a room is defined, it is placed in a floor plan for the house. The new customer then describes the next room or proceeds to the next action (which is to save the floor plan configuration). The movement across and down the sequence diagram ties each analysis class to use-case actions. If a use-case action

FIGURE 18.6 **Partial state diagram for new customer interaction**


New customer
Select "log-in"
Validating user
System status = "input ready"
Display msg = "enter userid"
Display msg = "enter pswd"
entry/log-in requested
do: run user validation
exit/set user access switch
userid validated
password validated
Selecting user action
System status = "link ready"
Display: "navigation choices"
entry/validated user
do: link as required
exit/user action selected
Select other functions
Customization complete
Select e-commerce (purchase) functionality
Select customization functionality
Next selection
Customizing
System status = "input ready"
Display: basic instructions
entry/validated user
do: process user selection
exit/customization terminated
Select descriptive content
Room being defined
All rooms defined
Defining room
System status = "input ready"
Display: room def. window
entry/room def. selected
do: run room queries
do: store room variables
exit/room completed
Select descriptive content
Saving floor plan
System status = "input ready"
Display: storage indicator
entry/room plan save selected
do: store floor plan
exit/save completed
Select save floor plan
Select descriptive content
Building floor plan
System status = "input ready"
Display: floor plan window
entry/floor plan selected
do: insert room in place
do: store floor plan variables
exit/room insertion completed
Select enter room in floor plan
Room insertion completed

is missing from the diagram, the Web engineer must re-evaluate the description of analysis classes to determine if one or more classes is missing. Sequence diagrams can be created for each use-case once analysis classes are defined for the use-case.

State diagrams. The UML state diagram (Chapter 8) provides another representation of the dynamic behavior of the WebApp as an interaction occurs. Like most modeling representations used in Web engineering (or software engineering), the state diagram can be represented at different levels of abstraction. Figure 18.6 depicts a partial, top-level (high level of abstraction) state diagram for the interaction between a new customer and the SafeHomeAssured.com WebApp.

In the state diagram shown, six externally observable states are identified: *validating user, selecting user action, customizing, defining room, building floor plan,* and *saving floor plan.* The state diagram indicates the events that are required to move the new customer from one state to another, the information that is displayed as a state is entered, the processing that occurs within a state, and the exit condition that causes a transition from one state to another.

Because use-cases, sequence diagrams, and state diagrams all present related information, it is reasonable to ask why all three are necessary. In some cases, they are not. Use-cases may be sufficient in some situations. However, use-cases