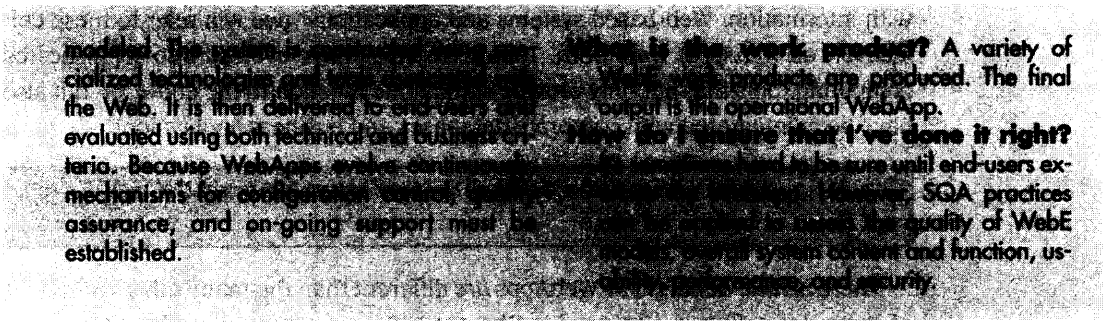


This leads us to a pivotal question: *Can software engineering principles, concepts, and methods be applied to Web development?* Many of them can, but their application may require a somewhat different spin.

But what if an undisciplined approach to Web development persists? In the absence of a disciplined process for developing Web-based systems, there is increasing concern that we may face serious problems in their successful development, deployment, and maintenance. In essence, the application infrastructure that we are creating today may lead to a "tangled Web" as we move further into this new century. This phrase connotes a morass of poorly developed Web-based applications that have too high a probability of failure. Worse, as Web-based systems grow more complex, a failure in one can and will propagate broad-based problems across many. When this happens, confidence in the entire Internet may be shaken. Worse, it may lead to unnecessary and ill-conceived government regulation, leading to irreparable harm to these unique technologies.

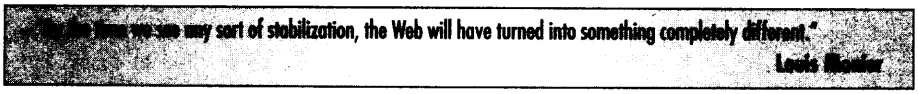
To avoid a tangled Web and achieve greater success in development and application of large-scale, complex Web-based systems, there is a pressing need for disciplined approaches and new methods and tools for development, deployment, and evaluation of Web-based systems and applications. Such approaches and techniques must take into account the special features of the new medium, the operational environments and scenarios, and the multiplicity of user profiles which pose additional challenges to Web-based application development.

Web engineering (WebE) applies "sound scientific, engineering, and management principles and disciplined and systematic approaches to the successful development, deployment and maintenance of high-quality Web-based systems and applications." [MUR99]

16.1 Attributes of Web-Based Systems and Applications

In the early days of the World Wide Web (circa 1990 to 1995), "Web sites" consisted of little more than a set of linked hypertext files that presented information using text and limited graphics. As time passed, HTML was augmented by development tools (e.g., XML, Java) that enabled Web engineers to provide computing capability along

with information. Web-based systems and applications[1] (we will refer to these collectively as *WebApps*) were born. Today, WebApps have evolved into sophisticated computing tools that not only provide standalone function to the end-user, but also have been integrated with corporate databases and business applications.



There is little debate that WebApps *are* different than the many other categories of computer software discussed in Chapter 1. Powell summarizes the primary differences when he states that Web-based systems "involve a mixture between print publishing and software development, between marketing and computing, between internal communications and external relations, and between art and technology." [POW98] The following attributes are encountered in the vast majority of WebApps.

It can be argued that a traditional application within any of the software domains discussed in Chapter 1 can exhibit this list of attributes. However, WebApps almost always do.

Network intensiveness. A WebApp resides on a network and must serve the needs of a diverse community of clients. A WebApp may reside on the Internet (thereby enabling open worldwide communication). Alternatively, an application may be placed on an Intranet (implementing communication across an organization) or an Extranet (inter-network communication).

Concurrency. A large number of users may access the WebApp at one time. In many cases, the patterns of usage among end-users will vary greatly.

Unpredictable load. The number of users of the WebApp may vary by orders of magnitude from day to day. 100 users may show up on Monday; 10,000 may use the system on Thursday.

Performance. If a WebApp user must wait too long (for access, for server-side processing, for client-side formatting and display), he or she may decide to go elsewhere.

Availability. Although expectation of 100 percent availability is unreasonable, users of popular WebApps often demand access on a "24/7/365" basis. Users in Australia or Asia might demand access during times when traditional domestic software applications in North America might be taken off-line for maintenance.

Data driven. The primary function of many WebApps is to use hypermedia to present text, graphics, audio, and video content to the end-user. In addition, WebApps are commonly used to access information that exists on databases that were

1 In the context of this chapter, the term "Web application" (WebApp) encompasses everything from a simple Web page that might help a consumer compute an automobile lease payment to a comprehensive Web site that provides complete travel services for business people and vacationers. Included within this category are complete Web sites, specialized functionality within Web sites, and information processing applications that reside on the Internet or on an Intranet or ExtraNet.

not originally an integral part of the Web-based environment (e.g., e-commerce or financial applications).

Content sensitive. The quality and aesthetic nature of content remains an important determinant of the quality of a WebApp.

Continuous evolution. Unlike conventional application software that evolves over a series of planned, chronologically spaced releases, Web applications evolve continuously. It is not unusual for some WebApps (specifically, their content) to be updated on a minute-by-minute schedule or for content to be independently computed for each request. Some argue that the continuous evolution of WebApps makes the work performed on them analogous to gardening. Lowe [LOW99] discusses this when he writes:

> Engineering is about adopting a consistent and scientific approach, tempered by a specific practical context, to development and commissioning of systems or applications. Web site development is often much more about creating an infrastructure (laying out the garden) and then "tending" the information which grows and blooms within this garden. Over time the garden (i.e., Web site) will continue to evolve, change, and grow. A good initial architecture should allow this growth to occur in a controlled and consistent manner. . . .

Continual care and feeding allows a Web site to grow (in robustness and importance). But unlike a garden, Web applications must serve (and adapt to) the needs of more than the gardener.

Immediacy. Although *immediacy*—the compelling need to get software to market quickly—is a characteristic of many application domains, WebApps often exhibit a time to market that can be a matter of a few days or weeks.[2] Web engineers must use methods for planning, analysis, design, implementation, and testing that have been adapted to the compressed time schedules required for WebApp development.

Security. Because WebApps are available via network access, it is difficult, if not impossible, to limit the population of end-users who may access the application. In order to protect sensitive content and provide secure modes of data transmission, strong security measures must be implemented throughout the infrastructure that supports a WebApp and within the application itself.

Aesthetics. An undeniable part of the appeal of a WebApp is its look and feel. When an application has been designed to market or sell products or ideas, aesthetics may have as much to do with success as technical design.

These general attributes apply to all WebApps, but with different degrees of influence.

---

2 With modern tools, sophisticated Web pages can be produced in only a few hours.

But what about the WebApps themselves? What problems do they address? The following application categories are most commonly encountered in WebE work [DAR99]:

What categories of WebApps are encountered in WebE work?

- *Informational*—read-only content is provided with simple navigation and links.
- *Download*—a user downloads information from the appropriate server.
- *Customizable*—the user customizes content to specific needs.
- *Interaction*—communication among a community of users occurs via chatroom, bulletin boards, or instant messaging.
- *User input*—forms-based input is the primary mechanism for communicating need.
- *Transaction-oriented*—the user makes a request (e.g., places an order) that is fulfilled by the WebApp.
- *Service-oriented*—the application provides a service to the user, e.g., assists the user in determining a mortgage payment.
- *Portal*—the application channels the user to other Web content or services outside the domain of the portal application.
- *Database access*—the user queries a large database and extracts information.
- *Data warehousing*—the user queries a collection of large databases and extracts information.

The attributes noted earlier in this section and the application categories noted above represent important facts of life for Web engineers. The key is living within the constraints imposed by these attributes and still producing a successful WebApp.

16.2 WebApp Engineering Layers

The development of Web-based systems and applications incorporates specialized process models, software engineering methods adapted to the characteristics of WebApp development, and a set of important enabling technologies. Process, methods, and technologies (tools) provide a layered approach to WebE that is conceptually identical to the software engineering layers described in Figure 2.1.

> "Web Engineering deals with disciplined and systematic approaches to development, deployment, and maintenance of Web-based systems and applications."
>
> Yogesh Deshpande

16.2.1 Process

WebE process models (discussed in detail in Section 16.3) embrace the agile development philosophy (Chapter 4). Agile development emphasizes a lean development

The WebE process is often agile and is almost always incremental. Note, however, that the agile model may not be chosen for major Web engineering projects.

approach that incorporates rapid development cycles. Aoyama [AOY98] describes the motivation for the agile approach in the following manner:

> The Internet changed software development's top priority from *what* to *when*. Reduced time-to-market has become the competitive edge that leading companies strive for. Thus, reducing the development cycle is now one of software engineering's most important missions.

Even when rapid cycle times dominate development thinking, it is important to recognize that the problem must still be analyzed, a design should be developed, implementation should proceed in an incremental fashion, and an organized testing approach must be initiated. However, these framework activities must be defined within a process that (1) embraces change, (2) encourages the creativity and independence of development staff and strong interaction with WebApp stakeholders, (3) builds systems using small development teams, and (4) emphasizes evolutionary or incremental development using short development cycles [MCD01].

16.2.2 Methods

The WebE methods landscape encompasses a set of technical tasks that enable a Web engineer to understand, characterize, and then build a high-quality WebApp. WebE methods (discussed in detail in Chapters 18 through 20) can be categorized in the following manner:

It's important to note that many WebE methods have been adopted directly from their software engineering counterparts. Others are in their formative stages. Some of these will survive; others will be discarded as better approaches are suggested.

Communication methods—define the approach used to facilitate communication between Web engineers and all other WebApp stakeholders (e.g., end-users, business clients, problem domain experts, content designers, team leaders, project managers). Communication techniques are particularly important during requirements gathering and whenever a WebApp increment is to be evaluated.

Requirements analysis methods—provide a basis for understanding the content to be delivered by a WebApp, the function to be provided for the end-user, and the modes of interaction that each class of user will require as navigation through the WebApp occurs.

Design methods—encompass a series of design techniques that address WebApp content, application and information architecture, interface design, and navigation structure.

Testing methods—incorporate formal technical reviews of both the content and design model and a wide array of testing techniques that address component-level and architectural issues, navigation testing, usability testing, security testing, and configuration testing.

It is important to note that although WebE methods adopt many of the same underlying concepts and principles as the software engineering methods described in Part 2 of this book, the mechanics of analysis, design, and testing must be adapted to accommodate the special characteristics of WebApps.

In addition to the technical methods that have just been outlined, a series of umbrella activities (with associated methods) are essential for successful Web engineering. These include project management techniques (e.g., estimation, scheduling, risk analysis), software configuration management techniques, and review techniques.

16.2.3 Tools and Technology

WebRef
Excellent resources for WebE technology can be found at **webdeveloper.com** and **www.eborcom.com/webmaker.**

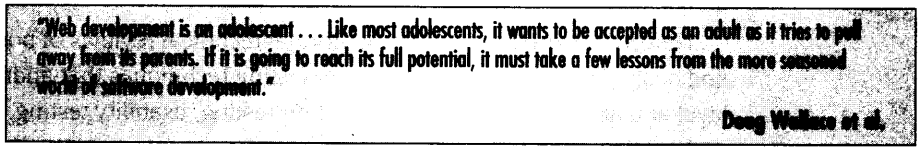
A vast array of tools and technology has evolved over the past decade as WebApps have become more sophisticated and pervasive. These technologies encompass a wide array of content description and modeling languages (e.g., HTML, VRML, XML), programming languages (e.g., Java) component-based development resources (e.g., CORBA, COM, ActiveX, .NET), browsers, multimedia tools, site authoring tools, database connectivity tools, security tools, servers and server utilities, and site management and analysis tools.

A comprehensive discussion of tools and technology for Web engineering is beyond the scope of this book. The interested reader should visit one or more of the following Web sites: *Web Developer's Virtual Encyclopedia* (www.wdlv.com), *WebDeveloper* (www.webdeveloper.com), *Developer Shed* (www.devshed.com), *Webknowhow.net* (www.webknowhow.net), or *WebReference* (www.webreference.com).

16.3 The Web Engineering Process

The attributes of Web-based systems and applications have a profound influence on the WebE process that is chosen. In Chapter 3 we noted that a software engineer chooses a process model based on the attributes of the software that is to be developed. The same holds true for a Web engineer.

If immediacy and continuous evolution are primary attributes of a WebApp, a Web engineering team might choose an agile process model (Chapter 4) that produces WebApp releases in rapid-fire sequence. On the other hand, if a WebApp is to be developed over a longer time period (e.g., a major e-commerce application), an incremental process model (Chapter 3) might be chosen.



The network intensive nature of applications in this domain suggests a population of users that is diverse (thereby making special demands on requirements elicitation and modeling) and an application architecture that can be highly specialized

(thereby making demands on design). Because WebApps are often content-driven with an emphasis on aesthetics, it is likely that parallel development activities will be scheduled within the WebE process and involve a team of both technical and non-technical people (e.g., copywriters, graphic designers).

16.3.1 Defining the Framework

Any one of the agile process models (e.g., Extreme Programming, Adaptive Software Development, SCRUM) presented in Chapter 4 can be applied successfully as a WebE process. The process framework that is presented here is an amalgam of the principles and ideas discussed in Chapter 4.

To be effective, any engineering process must be adaptable. That is, the organization of the project team, the modes of communication among team members, the engineering activities and tasks to be performed, the information that is collected and created, and the methods used to produce a high-quality product must all be adapted to the people doing the work, the project timeline and constraints, and the problem to be solved. Before we define a process framework for WebE, we must recognize that:

The WebE process model is predicated on three points: incremental delivery, continuous change, and short timelines.

1. *WebApps are often delivered incrementally.* That is, framework activities will occur repeatedly as each increment is engineered and delivered.
2. *Changes will occur frequently.* These changes may occur as a result of the evaluation of a delivered increment or as a consequence of changing business conditions.
3. *Timelines are short.* This mitigates against the creation and review of voluminous engineering documentation, but it does not preclude the simple reality that critical analysis, design, and testing must be recorded in some manner.

In addition, the principles defined as part of the "Manifesto for Agile Software Development" (Chapter 4) should be applied. However, the principles are not the Ten Commandments. It is sometimes reasonable to adopt the spirit of these principles without necessarily abiding by the letter of the manifesto.

With these issues in mind, we discuss the WebE process within the generic process framework presented in Chapter 2.

The generic process model (introduced in Chapter 2) is applicable to Web engineering.

Customer communication. Within the WebE process, customer communication is characterized by two major tasks: business analysis and formulation. *Business analysis* defines the business/organizational context for the WebApp. In addition, stakeholders are identified, potential changes in business environment or requirements are predicted, and integration between the WebApp and other business applications, databases, and functions is defined. *Formulation* is a requirements gathering activity involving all stakeholders. The intent is to describe the

problem that the WebApp is to solve (along with basic requirements for the WebApp) using the best information available. In addition, an attempt is made to identify areas of uncertainty and where potential changes will occur.

Planning. The project plan for the WebApp increment is created. The plan consists of a task definition and a timeline schedule for the time period (usually measured in weeks) projected for the development of the WebApp increment.

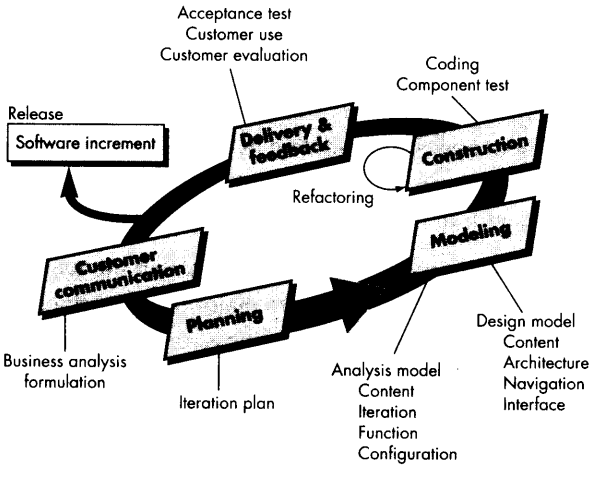
Modeling. Conventional software engineering analysis and design tasks are adapted to WebApp development, merged, and then melded into the WebE modeling activity (Chapters 18 and 19). The intent is to develop "rapid" analysis and design models that define requirements and at the same time represent a WebApp that will satisfy them.

Construction. WebE tools and technology are applied to construct the WebApp that has been modeled. Once the WebApp increment has been constructed, a series of rapid tests are conducted to ensure that errors in design (i.e., content, architecture, interface, navigation) are uncovered. Additional testing addresses other WebApp characteristics.

Deployment. The WebApp is configured for its operational environment, delivered to end-users, and then an evaluation period commences. Evaluation feedback is presented to the WebE team, and the increment is modified as required.

These five WebE framework activities are applied using an incremental process flow as shown in Figure 16.1.

FIGURE 16.1
The WebE process



INFO

Web Engineering—Basic Questions

The engineering of any product involves subtleties that are not immediately obvious to those without substantial experience. The characteristics of WebApps force Web engineers to answer a variety of questions that should be addressed during early framework activities. Strategic questions related to business needs and product objectives are addressed during formulation. Requirements questions related to features and functions must be considered during analysis modeling. Broad-based design questions related to WebApp architecture, interface characteristics, and navigational issues are considered as the design model evolves. Finally, a set of human issues, related to the manner in which a user actually interacts with the WebApp, are addressed on a continual basis.

Susan Weinshenk [WEI02] suggests a set of questions that must be considered as analysis and design progress. A small (adapted) subset are noted here:

- How important is a Web-site home page? Should it contain useful information or a simple listing of links that lead a user to more detail at lower levels?
- What is the most effective page layout (e.g., menu on top, on the right or left?), and does it vary depending upon the type of WebApp being developed?
- Which media options have the most impact? Are graphics more effective than text? Is video (or audio) an effective option? When should various media options be chosen?
- How much work can we expect a user to do when he or she is looking for information? How many clicks are people willing to make?
- How important are navigational aids when WebApps are complex?
- How complex can forms input be before it becomes irritating for the user? How can forms input be expedited?
- How important are search capabilities? What percentage of users browse, and what percent use specific searches? How important is it to structure each page in a manner that assumes a link from some outside source?
- Will the WebApp be designed in a manner that makes it accessible to those who have physical or other disabilities?

There are no absolute answers to questions such as these, and yet, they must be addressed as WebE proceeds. We'll consider potential answers in Chapters 17 through 20.

16.3.2 Refining the Framework

We have already noted that the WebE process model must be adaptable. That is, a definition of the engineering tasks required to refine each framework activity is left to the discretion of the Web engineering team. In some cases, a framework activity is conducted informally. In others, a series of distinct tasks will be defined and conducted by team members. In every case, the team has responsibility for producing a high-quality WebApp increment within the time period allocated.

It is important to emphasize that tasks associated with WebE framework activities may be modified, eliminated, or extended based on the characteristics of the problem, the product, the project, and the people on the Web engineering team.

"There are those of us who believe that the best practices for software development are practical and deserve implementation. And then there are those of us who believe that best practices are interesting in an academic sort of way, but are not for the real world, thank you very much."

Warren Keuffel

16.4 WEB ENGINEERING BEST PRACTICES

Will every WebApp developer use the WebE process framework and task set defined in Section 16.3? Probably not. Web engineering teams are sometimes under enormous time pressure and will try to take short-cuts (even if these are ill-advised and result in *more* development effort, not less). But a set of fundamental best practices—adopted from the software engineering practices discussed throughout Part 2 of this book—should be applied if industry-quality WebApps are to be built.

Be sure that the business need for a WebApp has been clearly enunciated by someone. If it hasn't, your WebE project is at risk.

1. *Take the time to understand business needs and product objectives, even if the details of the WebApp are vague.* Many WebApp developers erroneously believe that vague requirements (which are quite common) relieve them from the need to be sure that the system they are about to engineer has a legitimate business purpose. The end result is (too often) good technical work that results in the wrong system built for the wrong reasons for the wrong audience. If stakeholders cannot enunciate a business need for the WebApp, proceed with extreme caution. If stakeholders struggle to identify a set of clear objectives for the product (WebApp), do not proceed until they can.
2. *Describe how users will interact with the WebApp using a scenario-based approach.* Stakeholders must be convinced to develop use-cases (discussed throughout Part 2 of this book) to reflect how various actors will interact with the WebApp. These scenarios can then be used (1) for project planning and tracking, (2) to guide analysis and design modeling, and (3) as important input for the design of tests.
3. *Develop a project plan, even if it is very brief.* Base the plan on a predefined process framework that is acceptable to all stakeholders. Because project timelines are very short, schedule granularity should be fine; i.e., in many instances, the project should be scheduled and tracked on a daily basis.
4. *Spend some time modeling what it is that you're going to build.* Generally, comprehensive analysis and design models are not developed during Web engineering. However, UML class and sequence diagrams along with other selected UML notation (e.g., state diagrams) may provide invaluable insight.
5. *Review the models for consistency and quality.* Formal technical reviews (Chapter 26) should be conducted throughout a WebE project. The time spent on reviews pays important dividends because it often eliminates rework and results in a WebApp that exhibits high quality—thereby increasing customer satisfaction.
6. *Use tools and technology that enable you to construct the system with as many reusable components as possible.* A wide array of WebApp tools are available for virtually every aspect of WebApp construction. Many of these tools enable

a Web engineer to build significant portions of the application using reusable components.

7. *Don't rely on early users to debug the WebApp—design comprehensive tests and execute them before releasing the system.* Users of a WebApp will often give it one chance. If it fails to perform, they move elsewhere—never to return. It is for this reason that "test first, then deploy" should be an overriding philosophy, even if deadlines must be stretched.

INFO

Quality Criteria/Guidelines for WebApps

WebE strives to produce high-quality WebApps. But what is "quality" in this context, and what guidelines are available for achieving it? In his paper on Web-site quality assurance, Quibeldey-Cirkel [QUI01] suggests a comprehensive set of on-line resources that address these issues:

W3C: Style Guide for Online Hypertext
www.w3.org/Provider/Style
The Sevloid Guide to Web Design
www.sev.com.au/webzone/design/guide.asp
Web Pages That Suck
www.webpagesthatsuck.com/index.html
Resources on Web Style
www.westegg.com/unmaintained/badpages
Gartner's Web Evaluation Tool
www.gartner.com/ebusiness/website-ings
IBM Corp: Web Guidelines
www-3.ibm.com/ibm/easy/eou_ext.nsf/Publish/572
World Wide Web Usability
ijhcs.open.ac.uk
Interface Hall of Shame
www.iarchitect.com/mshame.htm
Art and the Zen of Web Sites
www.tlc-systems.com/webtips.shtml
Designing for the Web: Empirical Studies
www.microsoft.com/usability/webconf.htm
Nielsen's useit.com
www.useit.com
Quality of Experience
www.qualityofexperience.org
Creating Killer Web Sites
www.killersites.com/core.html
All Things at Web
www.pantos.org/atw
SUN's New Web Design
www.sun.com/980113/sunonnet
Tognazzini, Bruce: Homepage
www.asktog.com
Webmonkey
hotwired.lycos.com/webmonkey/design/?tw=design
World's Best WebSites
www.worldbestwebsites.com
Yale University: Yale Web-Style Guide
info.med.yale.edu/caim/manual

16.5 SUMMARY

The impact of Web-based systems and applications is arguably the single most significant event in the history of computing. As WebApps grow in importance, a disciplined WebE approach—adapted from software engineering principles, concepts, process, and methods—has begun to evolve.

WebApps are different from other categories of computer software. They are network intensive, content driven, and continuously evolving. The immediacy that drives their development, the overriding need for security in their operation, and the demand for aesthetic as well as functional content delivery are additional differentiating

factors. Like other types of software, WebApps can be assessed using a variety of quality criteria that include usability, functionality, reliability, efficiency, maintainability, security, availability, scalability, and time to market.

WebE can be described in three layers—process, methods, and tools/technology. The WebE process adopts the agile development philosophy that emphasizes a "lean" engineering approach that leads to the incremental delivery of the system to be built. The generic process framework—communication, planning, modeling, construction, and deployment—is applicable to WebE. These framework activities are refined into a set of WebE tasks that are adapted to the needs of each project. A set of umbrella activities similar to those applied during software engineering work—SQA, SCM, project management—apply to all WebE projects.

REFERENCES

[AOY98] Aoyama, M., "Web-Based Agile Software Development, *IEEE Computer,* November/December, 1998, pp. 56–65.

[DAR99] Dart, S., "Containing the Web Crisis Using Configuration Management," *Proc. First ICSE Workshop on Web Engineering,* ACM, Los Angeles, May 1999. (*The Proceedings of the First ICSE Workshop on Web Engineering* are published on-line at http://fistserv.macarthur.uws.edu.au/san/icse99-WebE/ICSE99-WebE-Proc/ default.htm).

[FOW01] Fowler M., and J. Highsmith, "The Agile Manifesto," *Software Development Magazine,* August 2001, http://www.sdmagazine.com/documents/s=844/ sdm0108a/0108a.htm.

[MCD01] McDonald, A., and R. Welland, *Agile Web Engineering (AWE) Process,* Department of Computer Science, University of Glascow, Technical Report TR-2001-98, 2001, downloadable from http://www.dcs.gla.ac.uk/~andrew/TR-2001-98.pdf.

[MUR99] Murugesan, S., *WebE Home Page,* http://fistserv.macarthur.uws.edu.au/ san/ WebEHome, July, 1999.

[NOR99] Norton, K., "Applying Cross Functional Evolutionary Methodologies to Web Development," *Proc. First ICSE Workshop on Web Engineering,* ACM, Los Angeles, May 1999.

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

[PRE98] Pressman, R. S. (moderator), "Can Internet-Based Applications Be Engineered?" *IEEE Software,* September 1998, pp. 104–110.

[QUI01] Quibeldey-Cirkel, K., "Checklist for Web Site Quality Assurance," *Quality Week Europe,* 2001, downloadable from www.fbi.fh-darmstadt.de/~quibeldey/ Projekte/QWE2001/ Paper_Quibeldey_Cirkel.pdf.

[WEI02] Weinschenk, S., "Psychology and the Web: Designing for People," 2002, http://www.weinschenk.com/learn/facts.asp.

PROBLEMS AND POINTS TO PONDER

16.1. Using an actual Web site as an example, illustrate the different manifestations of WebApp "content."

16.2. Do a bit of research and write a two to three page paper that summarizes one of the technologies noted in Section 16.2.3.

16.3. How do you judge the "quality" of a Web site? Make a prioritized list of 10 quality attributes that you believe are most important.

16.4. Are there other generic attributes that differentiate WebApps from more conventional software applications? Try to name two or three.

16.5. Review the discussion of the "Manifesto for Agile Software Development" presented in Chapter 4. Which of the 12 principles would work well for a two-year project (involving dozens of people) that will build a major e-commerce system for an automobile company? Which of the 12 principles would work well for a two-month project that will build an informational site for a small real estate firm?

16.6. Make a list of "risks" that would be likely during the development of a new e-commerce application that is designed to sell mobile phones and service directly over the Web.

16.7. Review the software engineering processes described in Chapter 3 and 4. Is/are there another process(es)—other than the agile process model—that might be applicable to Web engineering? If yes, indicate which process(es) and why.

Further Readings and Information Sources

Hundreds of books that discuss one or more Web engineering topics have been published in recent years, although relatively few address all aspects of WebE. Sarukkai (*Foundations of Web Technology,* Kluwar Academic Publishers, 2002) presents a worthwhile compilation of the technologies that are required for WebE. Murugusan and Deshpande (*Web Engineering: Managing Diversity and Complexity of Web Development,* Springer-Verlag, 2001) have edited a collection of useful papers on WebE. Proceedings of international conferences on Web Engineering and Web Information Systems Engineering are published yearly by the IEEE Computer Society Press.

Flor (*Web Business Engineering,* Addison-Wesley, 2000) discusses business analysis and related concerns that enable the Web engineer to better understand customer needs. Bean (*Engineering Global E-Commerce Sites,* Morgan Kaufmann, 2003) presents guidelines for the development of global WebApps. Lowe and Hall (*Hypermedia and the Web: An Engineering Approach,* Wiley, 1999) and Powell [POW98] provide reasonably complete coverage. Umar (*Application Re-engineering: Building Web-Based Applications and Dealing with Legacy Systems,* Prentice-Hall, 1997) addresses one of the most difficult issues in WebE—the re-engineering of legacy systems to make them compatible with Web-based systems. IEEE Std. 2001-1999 defines basic WebE practices.

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

CHAPTER

17 INITIATING A WEBAPP PROJECT

KEY CONCEPTS

- analysis types
- communication
- e-projects
- formulation questions
- goals and objectives
- in-house WebE
- metrics
- outsourcing
- planning
- project management
- requirements gathering
- teams
- worst practices

During the roaring 1990s, the Internet boom generated more hubris than any other event in the history of computers. WebApp developers at hundreds of young dot.com companies argued that a new paradigm for software development had arisen, that old rules no longer applied, that time-to-market trumped all other concerns. They laughed at the notion that careful formulation and planning should occur before construction commenced. And who could argue? Money was everywhere, 24-year olds became multimillionaires (on paper, at least)—maybe things really had changed. And then the bottom fell out.

It became painfully apparent as the twenty-first century began that a "build it and they will come" philosophy just doesn't work, that problem formulation is essential to ensure that a WebApp is really needed, and that planning is worth the effort, even when development schedules are tight. Constantine and Lockwood [CON02] note this situation when they write:

> Despite breathless declarations that the Web represents a new paradigm defined by new rules, professional developers are realizing that lessons learned in the pre-Internet days of software development still apply. Web pages are user interfaces, HTML programming is programming, and browser-deployed applications are software systems that can benefit from basic software engineering principles.

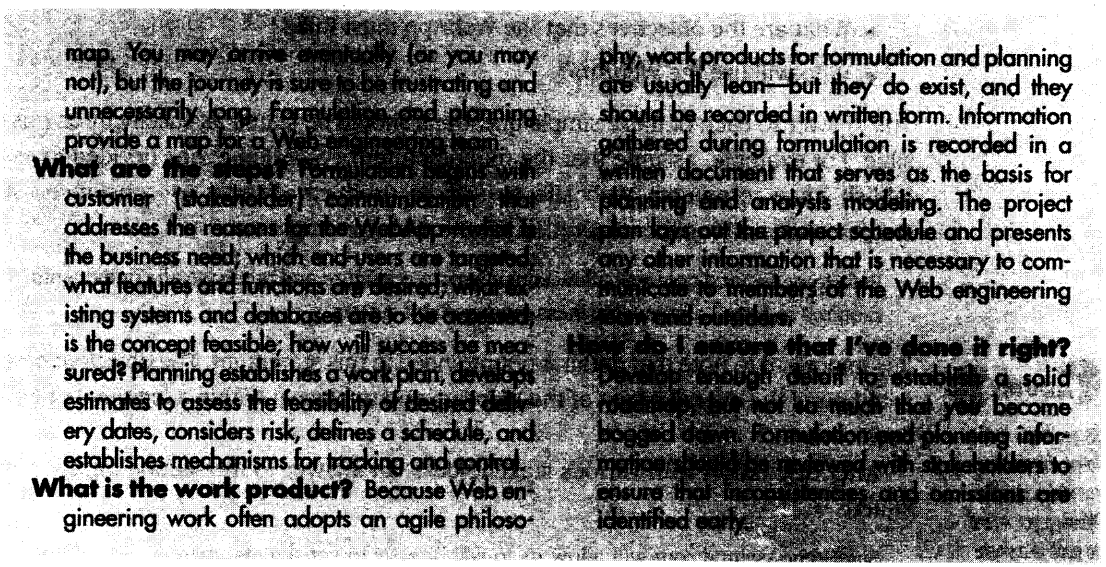
Among the most fundamental principles of software engineering is: *Understand the problem before you begin to solve it, and be sure that the solution you conceive is one that people really want.* That's the basis of formulation, the first major activity in Web engineering. Another fundamental software engineering principle is: *Plan the work before you begin performing it.* That's the philosophy that underlies project planning.

QUICK LOOK

What is it? Getting started is always difficult. On one hand, there is a tendency to procrastinate, to wait until every *t* is crossed and every *i* is dotted before work begins. On the other hand, there is a desire to jump right in, to begin building even before you really know what needs to be done. Both approaches are inappropriate, and that's why the first two Web engineering framework activities emphasize formulation and planning. Formulation assesses the underlying need for the WebApp, the overall features and functions that users desire, and the scope of the development effort. Planning addresses the things that must be defined to establish a work flow and a schedule, and to track work as the project proceeds.

Who does it? Web engineers, their managers, and nontechnical stakeholders all participate in formulation and planning.

Why is it important? It's hard to travel to a place you've never visited without directions or a


map. You may arrive eventually (or you may not), but the journey is sure to be frustrating and unnecessarily long. Formulation and planning provide a map for a Web engineering team.

What are the steps? Formulation begins with customer (stakeholder) communication that addresses the reasons for the Web development, the business need, which end-users are targeted, what features and functions are desired, which existing systems and databases are to be accessed, is the concept feasible, how will success be measured? Planning establishes a work plan, develops estimates to assess the feasibility of desired delivery dates, considers risk, defines a schedule, and establishes mechanisms for tracking and control.

What is the work product? Because Web engineering work often adopts an agile philosophy, work products for formulation and planning are usually lean—but they do exist, and they should be recorded in written form. Information gathered during formulation is recorded in a written document that serves as the basis for planning and analysis modeling. The project plan lays out the project schedule and presents any other information that is necessary to communicate to members of the Web engineering team and customers.

How do I ensure that I've done it right? Develop enough detail to establish a solid foundation, but not so much that you become bogged down. Formulation and planning information should be reviewed with stakeholders to ensure that inconsistencies and omissions are identified early.

17.1 FORMULATING WEB-BASED SYSTEMS

Formulation of Web-based systems and applications represents a sequence of Web engineering actions that begins with the identification of business needs, moves into a description of WebApp objectives, defines major features and functions, and performs requirements gathering that leads to the development of an analysis model. Formulation allows stakeholders and the Web engineering team to establish a common set of goals and objectives for the construction of the WebApp. It also identifies the scope of the development effort and provides a means for determining a successful outcome. Analysis—a technical activity that is a continuation of formulation—identifies the data, functional, and behavioral requirements for the WebApp.

Formulation focuses on the "big picture"—on business needs and objectives and related information.

Before we consider formulation in more detail, it is reasonable to ask where formulation stops and requirements analysis begins. There is no easy answer to this question. Formulation focuses on the "big picture"—on business needs and objectives and related information. However, it is virtually impossible to maintain this level of abstraction. Stakeholders and Web engineers want to define required content, discuss specific functionality, enumerate specific features, and identify the manner in which end-users will interact with the WebApp. Is this formulation or requirements gathering? The answer is both.

17.1.1 Formulation Questions

Powell [POW98] suggests a set of questions that should be asked and answered at the beginning of the formulation step:

- What is the main motivation (business need) for the WebApp?

- What are the objectives that the WebApp must fulfill?
- Who will use the WebApp?

The answer to each of these simple questions should be stated as succinctly as possible. For example, assume that the manufacturer of *SafeHome*[1] has decided to establish an e-commerce Web site to sell its products directly to consumers. A statement describing the motivation for the WebApp might be:

> SafeHomeAssured.com will allow consumers to configure and purchase all components required to install a home/business management system.

As you begin formulating the problem, try to describe the WebApp you intend to build in a single sentence. If you can't, you don't understand the overall goals of the work.

It is important to note that detail is not provided in this statement. The objective here is to bound the overall intent of the WebApp and to place it within a legitimate business context.

After discussion with various stakeholders, an answer to the second question is stated:

> SafeHomeAssured.com will allow us to sell directly to consumers, thereby eliminating middleman costs and improving our profit margins. It will also allow us to increase sales by a projected 25 percent over current annual sales and will allow us to penetrate geographic regions where we currently do not have sales outlets.

Finally, the company defines the demographic for the WebApp: "Projected users of SafeHomeAssured.com are homeowners and owners of small businesses."

The answers stated above imply specific goals for the SafeHomeAssured.com Web site. In general, two categories of goals [GNA99] are identified:

- *Informational goals*—indicate an intention to provide specific content and/or information for the end-user.
- *Applicative goals*—indicate the ability to perform some task within the WebApp.

In the context of the SafeHomeAssured.com WebApp, one informational goal might be:

> The site will provide users with a detailed product specification, including technical descriptions, installation instructions, pricing information.

Examination of the answers to the questions posed above might lead to the statement of an applicative goal:

> SafeHomeAssured.com will query the user about the facility (i.e., house, office/retail space) that is to be protected and make customized recommendations about the product and configuration to be used.

Once all informational and applicative goals have been identified, a user profile is developed. The user profile captures "relevant features related to potential users

---

1 The *SafeHome* product has been used as an example throughout Parts 1 and 2 of this book.