Web-Based De Complex Inform

ROY T. FIELDING,
E. JAMES WHITEHEAD, JR.,
KENNETH M. ANDERSON,
GREGORY A. BOLCER,
PEYMAN OREIZY,
AND RICHARD N. TAYLOR

Want to launch a virtual enterprise using the Web as your technology infrastructure? First learn what's missing from today's Web architecture and which extensions are needed for large-scale collaboration.

The World-Wide Web connects islands of information, along with the people seeking that information, from within corporate intranets and across the global Internet, easily and effectively. Sharing information without regard to physical location has prompted new forms of virtual business and social endeavors. A virtual enterprise is an organization unconstrained by geographic location, and a membership intersecting multiple traditional organizations. Virtual enterprises can be formed within large corporations (consisting of groups at distributed sites), as parts of business alliances or task forces [4], and even among individuals working independently of any corporate connection. Indeed, all that is needed to form a virtual enterprise is at least one common goal, a shared information space, a means of coordinating users' efforts, and people willing to share the work.

The Web provides the minimum for setting up such enterprises by enabling identification of shared goals and the people who share them, by providing a standard mechanism for reading the shared information space, and by supporting coordination via email archives. However, the existing Web support infra-

velopment of ation Products

structure requires specialized software installation and nonstandard interfaces, and remains paltry in regard to remote authoring and task coordination. Our purpose here and in our work at the University of California, Irvine, is to identify those aspects of the Web infrastructure that need improvement and recommend specific ways to improve them in order to realize the Web's full potential for all forms of virtual enterprise.

A typical goal for virtual enterprises is development and maintenance of a complex information product we define as a highly interconnected and interdependent package of information. Example information products include a gourmet recipe collection, a book or journal article with multiple authors, an advertising campaign, or the technical documentation for some other complex product, such as a maintenance manual for an airplane built from components manufactured around the globe. Although the focus of our research and of this article is the example of a software engineering product, the lessons described and the solutions suggested are applicable to all forms of complex information products and thus are usable in all forms of virtual enterprise for which an information product is a goal.

Software Engineering As a Virtual Enterprise

Software engineering is fundamentally about the production and maintenance of complex information products. The Web has the potential for providing the infrastructure of a global software engineering environment, seamlessly supporting evolution of a software product from concept through implementation, deployment, and long-term maintenance, regardless of location or number of people involved. Such an environment supports the addition, subtraction, and migration of participants in the software process without requiring changes to the infrastructure, thus limiting the effect of changes on the project as a whole and providing a continuous source of information for users of the software products. Such an environment is precisely what is needed to support the communication and coordination of a virtual enterprise, particularly when the enterprise focuses on creating a complex information product.

The full benefit of a Web-based enterprise is not apparent until one considers the options available at each point in the product development process. The exchange of ideas and the encouragement of participants can involve colleagues around the world. Notes, diagrams, and sketches are available to any-

Goal	Enabling technology missing from the Web
Linking all artifacts and processes (people and tasks)	Links as first-class objects and a client architecture for hypermedia communication between viewers of a multitude of data formats
Flexible interaction model and hypermedia services	Component-based client architecture with hypermedia workspace manager and data-specific handlers; notification services
Distributed annotation	Remote linking and links as first-class objects
Visibility of artifacts over time	Versioning of resources
Distributed authoring	Remote locking, linking, access control, and versioning of resources
Distributed coordination and change management	All of the above

Table 1. Virtual organization goals and related supporting technology currently not available from the Web

one within the enterprise and accessible anytime, so consideration of their value is not limited to meetings with their authors. All project materials can be examined, searched, and edited online; other benefits include automated support for versioning (retrieval of older editions of an artifact), establishment of hypermedia relationships among work products and people and tasks, and the ability to track the related dependencies as they change over time. Plans, designs, specifications, and code can be located and reused, partly or completely, from earlier projects. Finally, end users are not left out of the process; they can contribute directly to development, limited only by the access restrictions assigned by the development organization.

An example of how the Web can help software engineering projects is the Apache Hypertext Transfer Protocol (HTTP) server project [7], founded in 1995 by a group of Webmasters to ensure the continued development of a freely available server implementation of HTTP [6]. Starting with a core group of eight volunteers, the Apache server has been iteratively designed and developed as a collaborative project, using only email and the Web for project communication and coordination and product distribution. The Apache server is now running on more than 50% of all Internet Web sites (more than 1 million hosts), competing successfully against multi-billion-dollar corporations. However, the Apache Group collaborates effectively across the Internet because the people involved are all Webmasters—experts in network administration, installation of new software, and communication via email, FTP, and HTTP—and thus do their own tool integration and coordination work, which is beyond the abilities of most developers. However, the Apache Group has encountered obstacles it has had to work around manually, resulting in frustration and delays. Our goal is to make the technology for remote collaboration and coordination a standard part of the Web infrastructure, so the Web itself becomes an environment for global software engineering and virtual enterprises in general, and thus the ability to work within a virtual enterprise will always be available on a user's desktop. Table 1 summarizes the changes we recommend for the Web infrastructure and the opportunities they would make possible.

Our goal is to make the technology for remote collaboration and coordination a standard part of the Web infrastructure.

First-Class Links

The core design consideration of any hypermedia system is how it models relationships as links [8]. The Web today defines links according to the media type of each node (document), with HTML being the dominant media type. Links in HTML are unidirectional and embedded directly within a document's markup. This approach is the simplest method for implementing links, and since the links are distributed with each document, it performs well in terms of scalability and disconnected operation. However, the embedded link model generally hinders the ability to perform link maintenance and prevents the creation of links over documents that are read-only or in a data type not hypermedia-aware.

Link maintenance is a primary concern for distributed hypermedia systems like the Web, in which the nodes being linked may be owned and changed independently of the nodes linking to them. If the destination of a link is moved or deleted, the source becomes a dangling link until it is updated. In response to this problem, site management tools like MOMspider [5] aid Webmasters in maintaining anchor references. Unfortunately, these tools are hindered in their support for intersite management due to the lack of standard mechanisms for notifying remote sites about changes to local sites and by their inability to directly edit the source documents to update the links automatically.

Open hypermedia systems have demonstrated that making links first-class objects—separating the

link definitions from the hypermedia content and providing an interface for link manipulation—solves many of these problems. The separation provided by this approach allows the links and anchors of a complex information product to be manipulated and analyzed more easily. Changes to these structures can be made independently of changes to hypermedia content, because they are stored separately, and the external representation allows related nodes to be updated automatically. In addition, this separation enables creation of anchors and links in existing documents, even when access to these documents is read-only. Annotation is readily supported by this approach, because making an annotation is equivalent to defining a link from an area of an existing document to a separate note. Likewise, the link server interface makes it easier for tools to automatically generate overviews of hypermedia content and guided tours. First-class links are typically modeled as sets, enabling links with more than a single destination (*n*-ary links). Furthermore, typing can be applied to links, allowing a variety of relationships to be defined with distinct run-time semantics and the ability to search for specific link types.

However, separating links from hypermedia content has drawbacks. Access to the external links becomes dependent on a particular link server (in which the links are stored), and disconnected operation is not possible without copying the associated external links to a local link server. Attempting to maximize the availability of links leads to scalability problems due to the potential desire of Webmasters to centralize control over shared sets of external links and the need for distributed link servers to be aware of each other and to communicate updates. Furthermore, any change to the hypermedia content made by non-hypermedia-aware tools will result in the same dangling-link problems as in the Web, if not worse problems due to the loss of consistency between content and the separate anchor specifications. However, the value of the features enabled by first-class links far exceeds their risks. An appropriate hybrid design would enable first-class links without adversely affecting existing Web content and embedded links.

Several approaches augment Web applications with first-class links [1]. For example, clients could be modified to access an open-hypermedia-system service in parallel with each Web request. Alternatively, each Web request could be filtered through a link server using HTTP's proxy support, as demonstrated by Microcosm [3] and OzWeb [9], allowing the link server to dynamically wrap or embed first-class links within the normal HTTP response. However, these methods are inefficient, motivated primarily by the desire of open hypermedia systems to work with existing Web clients and servers. A more drastic approach is to replace the Web protocols entirely, as has been done with Hyper-G [11], and provide substandard interfaces to clients using the older protocols. The best, approach over the long term is to use HTTP's existing extensibility mechanisms and incorporate the ability to identify link servers and transfer first-class links directly within the standard protocol, enabling future Web applications to take full advantage of these abilities without requiring a separate interface for older applications.

Notification

HTTP, the Web's primary information transfer protocol, is based on a strict client/server model. A typical HTTP server waits for client requests, locates the requested resource, applies the requested method to that resource, and sends the response back to the client. Although this model of communication scales well for simple retrieval tasks, it is not sufficient for the complex interactions in software engineering (or in any collaborative work process). A change in one resource often necessitates other changes to maintain dependencies between resources. In a strict client/server model, the client has to poll the server for changes to a resource, but polling is extremely inefficient when the resource space is large or when changes are infrequent. Needed is a means for clients to register interest in a resource and for servers to supply a notification when the resource changes.

Notification is not an entirely new concept for the Web; third-party services monitor a given resource (usually by periodic polling) and send email as notification when the resource changes. However, registering for such services is a manual process, as is receipt and processing of the email responses. These services improve efficiency only in terms of reducing the number of clients performing the polling; the Web needs greater flexibility in terms of the client specifying the protocol and message format of the notification, since the message granularity and delivery requirements vary by type of application and frequency of change.

Support for notifications could be added within the HTTP protocol as a form of first-class link. A server supporting notification could observe a change to the resource, check the resource for links of type "notify," and post a notification message to the link's destination (identified by a URL) in a format indicated by the link attributes. Notification is therefore enabled once support for remote link authoring services is added to the Web.

Client Architecture

Software engineering involves a multitude of specialized data formats—source code, specifications, test results, project plans, design diagrams, and more—each introducing important relationships and dependencies within an overall project. One of our goals is to be able to manipulate all of this data as hypermedia, including adding anchors and linking relationships to the objects represented within each data type, rather than to just an overlay of a particular rendition of that data. We need data-specific handlers (viewers, editors, and other tools) with equal access to hypermedia functionality, allowing modes of interaction that take advantage of the properties of each particular data type.

One example of how data-specific handlers can improve hypermedia functionality is by enabling implicit links derived from the nature of an artifact rather than being explicitly defined by an anchor or external link specification. For example, if a program is written in the C programming language and we have an indexed hypertext language reference manual for C, there is an implicit relationship between every C keyword and operator in the program and its corresponding definition in the language reference manual (LRM). While it is possible to explicitly instantiate every one of these relationships as an independent link, it is more efficient to define the abstract relationship

{ keyword } —> http://site/LRM?keyword

and allow the actual link to be calculated only when invoked by the user. Other implicit links commonly found in source code include *def*inition-*use* relationships, begin-end bracketing, and next-statement jumps, each of which can be calculated by a viewer with knowledge of the source code language similar to that of a compiler.

The core design consideration of any hypermedia system is how it models relationships as links.

Current Web clients use a number of mechanisms to allow hypermedia interaction with data formats other than HTML. The most basic is the media-type handler (sometimes called the mimecap interface) consisting of a program to execute when a particular data type is retrieved. Although media-type handlers are the basis for most solutions, it does not by itself include any hypermedia-aware interface, and thus the handler has to invoke an additional interface to do anything more than act as a read-only window. Current forms of this additional hypermedia interface include the Inter-Client Communication Protocol (ICCP) and the Netscape plug-in mechanism. Though useful, these interfaces do not support the full range of hypermedia functionality and require the constant presence of the primary browser application.

Coordinated tool interaction and the hypermedia workspace. Another mechanism for data-specific handlers involves applets that supply the rendering and manipulation code for a specific data type. Although applets often provide a hypermedia interface, these interfaces are secluded from the overall hypermedia workspace for security reasons, and thus two or more applets are generally prevented from cooperating on a single task. Applets need to be able to register interfaces restricted to safe interactions, so the security constraints of a given application are enforced with greater precision than a blanket prohibition.

The Web client architecture has traditionally been dominated by the monolithic browser, a huge application acting as window manager, hypermedia viewer, network request controller, and manager of user preferences, bookmarks, and history. It is difficult for anyone to introduce new functions to such an architecture, particularly for the multitude of data-specific handlers needed for software engineering. The client architecture has to be replaced by an architecture consisting of a dynamic collection of small, communicating applications, similar to the way Apple Computer's Cyberdog client consists of a collection of OpenDoc components. However, lacking today is the glue—the interface specifications—

that can hold these components together to form a consistent hypermedia workspace.

Software engineers use a diverse collection of tools to support their activities, including editors, debuggers, version control systems, and static and dynamic analyzers, that are not always operated in isolation; rather, multiple tools are used together to solve a particular task. However, tool coordination frequently lacks an interface mechanism. Current monolithic browsers serve that function internally but only within the limited scope of their original design. The problem is that in order to provide for more flexible and extensible clients, the components of the client need to be independent, yet they cannot work effectively as a hypermedia workspace unless something unifies their behavior in response to hypermedia events and user actions. In other words, a hypermedia workspace manager has to provide a standard set of services that components can access to register handlers and initiate external hypermedia events.

For example, consider the communication patterns of cooperating code editor, design viewer, and run-time debugger tools. When executing, the run-time debugger acts as a traversal engine; the link being traversed is the implicit one between a completed code statement and the next statement as determined by the program control flow. Each of these traversals can be viewed as a hypermedia event, and the code editor and the design viewer can both register interest in these events, perhaps with differing granularity, in much the same way a hypertext history-mapping tool registers interest in the traversals of a typical browser. Likewise, the user may wish to set a breakpoint by selecting a module in the design viewer or a statement in the code viewer. Such complex interaction is possible only in a hypermedia-based software environment if the component architecture does not artificially constrain the hypermedia interface to actions typical of traditional browser applications.

Distributed Authoring and Versioning

An essential element differentiating a virtual enterprise from a traditional organization is the likelihood that the participants are distributed across multiple, remote locations. Likewise, all virtual enterprises need at least one shared information space—for recording decisions, exchanging ideas, and storing work products. If the participants are distributed, they need a mechanism for distributed write-access to the shared space. Even when the enterprise begins as a local project, the advantage of a system supporting later distribution is that the participants do not need to change their working environment if the project expands later. This scalability from local to remote use is a notable advantage for virtual enterprises engaged in software engineering, since a successful software product is almost always used beyond the original development team, and it is often difficult to determine just how successful a product will be before much of the original design information is lost.

HTTP provides the bare essentials for distributed authoring of a virtual organization's Web-based content, with its PUT (write a resource) method and entity tags [7]. Unfortunately, these abilities do not represent the complete set of features required by users when performing distributed authoring. Existing HTML authoring tools supporting a remote "publish" operation often use custom extensions to HTTP or a manually operated Common Gateway Interface (CGI, a standard for running external programs from a Web HTTP server) to meet their needs. To date, the vast majority of authoring practices assume direct access (typically via a file system) to the underlying storage medium for Web content. When direct access is not available, as in remote authoring, many of the Web's innate weaknesses become evident, including the inability to version a resource, get a directory listing, make a new directory, copy or move a resource, set attributes, or create relationships between resources.

Authentication and access control. A prerequisite to enabling distributed authoring and versioning is the ability to perform authentication, by which the recipient of a message verifies the identity of the sender (both as an individual and in terms of his or her group memberships), and access control, which associates a request for the particular resource that is the object of the request with the people allowed to make that type of request. The Web provides some weak forms of authentication suitable only for a controlled network environment; access control is generally defined within the internal configuration of Web servers. We hope that stronger forms of authentication will be developed and adopted by the general Web community. (Many solutions have already been proposed.) Once authentication is enabled, remote authoring of access control configurations will also be possible.

Locking. A common problem with remote authoring, called the lost update problem, occurs when two people collaborating on a single document overwrite each other's work in successive requests. One solution is to temporarily exclude access to a resource; if a resource is to be edited, the author first

requests a write lock, thus preventing others from writing to the same resource. Such long-term pessimistic locking necessitates the ability to query a resource for its current locks and provides a means for lock management. An alternative solution, called short-term optimistic locking, is to copy the resource(s) being edited to a separate workspace, perform the editing within that workspace, and then lock the resource during the short period required to commit all the changes at once. Both solutions require verification that the resource has not changed during editing, as well as the ability to obtain a write-lock during the period between the beginning of the verification process and the completion of the edits.

Versioning. The Web today supports only one version of a document—the current one. However, saving and retrieving past versions of a document are critical in software development. The ability to store and access previous document versions, retrieve the history of a document, branch and merge revision paths, annotate document revisions with comments about the changes, and retrieve information about the differences among document versions are all useful in any shared information space, and thus for any virtual enterprise.

Collections. The Web today gives inadequate support for collections, or groupings, of related resources. Collections can be used to organize the resource namespace, automate guided tours through a set of resources, represent version histories, and enable moving, copying, and deletion of multiple resources in a single action. An immediate need for collections is in remote authoring applications requiring support for a `Save As...` dialog box providing a list of the current contents of a hierarchical region of the resource namespace in much the same way a file system directory provides a list of filenames. Collections of resources on the Web would not be limited to operating system directories, a single resource might belong to multiple collections, and a collection could include resources from multiple sites.

Copy and move. File systems, configuration management systems, and document management systems all permit users to copy documents and change document names without altering their contents. There are compelling reasons for the incorporation of this functionality into the Web as well. A copy function can be used to duplicate a document before a modification sequence is started. A move function allows expansion of naming conventions as a project grows. Names acceptable for a small set of documents are often too informal for a larger set; a move function bridges the old and new naming schemes. While these operations could be supported by the Web as it is today by loading and resaving the contents of a document, this loading and resaving operation is extremely inefficient for large documents and recursive copies (such as copying a URL hierarchy).

Interfaces must allow a user's favorite tools to become part of the hypermedia workspace.

Metadata. Resources on the Web consist of both data representations and metadata, or information describing the attributes of the representations. Although HTTP supports the transfer of metadata as message header fields, distributed authoring has to be able to create, modify, and delete metadata. Providing simple attribute-value pairs would allow the recording of many valuable aspects of a document (such as author, title, subject, organization, and keywords). These attributes have many uses, including support for searches on attribute contents and creation of catalog entries as placeholders for documents not yet available in electronic form. In addition, first-class links and externally specified anchors are most naturally implemented as metadata within HTTP, so distributed authoring of first-class links requires the same distributed authoring of metadata.

All of these enhancements to HTTP and the Web infrastructure are being pursued through the Internet Engineering Task Force's working group on World-Wide Web Distributed Authoring and Versioning (see www.ics.uci.edu/pub/ietf/webdav) [12].

Coordination

The most significant problems caused by the distributed nature of virtual enterprises involve coordination. Malone and Crowston [10] define coordination as "the act of managing interdependen-

cies between activities performed to achieve a goal." Because participants may not be in the same geographic location and may have working schedules that never overlap, people in virtual enterprises cannot observe and anticipate the factors that affect the interdependencies among tasks. A virtual enterprise typically lacks opportunities for informal conversations, like those in traditional organizations during coffee breaks and hallway exchanges, that help provide the big picture needed by employees and management alike to anticipate coordination problems. Likewise, the time cost in asking what everyone is working on (to avoid duplicated effort) may exceed the time it takes to simply perform the task at hand.

Coordination within a virtual enterprise is characterized by several modes:

- Matching resources, including people, equipment, and documents, with tasks and negotiating their roles within the tasks
- Forming and identifying constraints, responsibilities, deliverables, plans, and interdependencies
- Carrying out the process, including scheduling, handoff, and sharing of data
- Establishing completion criteria

These modes occur continuously throughout an enterprise, with many iterations and at varying levels of task granularity.

The Web supports limited coordination through provision of shared information spaces. But to fully participate in a virtual enterprise, people need to be able not only to exchange data but to negotiate the policies governing their collaboration. This negotiation requires a task-oriented view of the project, rather than just the data-oriented view provided by the Web. Virtual enterprises like the Apache Group generally use email lists and manually updated agendas to minimally support their coordination activities. Although we can build tools to support a task-oriented view, implementing these tools on top of the Web infrastructure first requires implementation of the improvements we've cited in this article.

Support for matching resources with tasks and for negotiating the roles within tasks requires distributed authoring and annotation capabilities. Although the Web supports distribution of read-only agendas, task and interest lists, and similar forms of organizational data, such data is useful for coordination only when it is kept up to date, mirroring the enterprise's actual state. Keeping up in turn requires equal access and unfiltered input from those participating or desiring to participate in the tasks.

Formation and identification of constraints, responsibilities, deliverables, plans, and interdependencies requires the linking of people, tasks, processes, and the artifacts created by these processes. A hypermedia system can effectively model the interdependencies within a virtual enterprise but only if the resources used and created by the enterprise are represented as nodes within the hypermedia system. Although this type of modeling was part of Berners-Lee's original vision for how the Web would be used within organizations [2], the infrastructure to support it has been neglected. Hypermedia capabilities need to be equally available to formats other than HTML, with data-specific handlers for operating on those formats—implying the need for first-class links and component-based client architectures. Likewise, accurate planning and identification of existing resources often require the continued visibility of older versions of artifacts and historical plans, explaining the need for the versioning of resources.

Support for execution of the process requires a flexible interaction model and hypermedia services unlimited by the traditional role of read-only browsers. Participants in the process cannot be expected to give up their favorite editors, compilers, debuggers, or analogous tools in exchange for an interface emasculated for the sake of genericness. Instead, interfaces must allow a user's favorite tools to become part of the hypermedia workspace.

Finally, establishing completion criteria requires notification services. Completion of an activity today can be indicated only by the appearance of a new link, creation of a new artifact, or some external notification mechanism, such as email. Other participants might obtain status information by polling for the existence of the artifact or link but become entangled in a "keep checking back" syndrome. With notification support built into the Web via remote link authoring and first-class links, an event-driven, task-oriented model is possible; participants can be notified directly of key project events, as needed and when appropriate. Similar to the way some systems allow controlled exchanges by "granting the floor" to the current speaker, such events could be broadcast, synchronized, and scheduled to create policies for controlled information sharing between distributed sites.

Conclusion

Just as the Web eliminated the barriers to personal publishing, virtual enterprises have the potential to eliminate many barriers between people working

toward a common goal. If initiating a collaborative activity becomes as simple as exchanging authentication credentials, anyone with access to a Web server could create a project with global scope. And, due to the number of people with access to the Web, almost any project could attract a sufficient number of people sharing the same goal. Naturally, none of these potential benefits will be realized easily; there are many technical and social pitfalls associated with improving the infrastructure of a system as large as the Web. Nevertheless, we already see the powerful abilities of virtual enterprises in examples like the Apache project. People's natural desire to collaborate with one another will only increase as new opportunities for collaboration develop.

The infrastructure improvements we described are applicable to any virtual enterprise that includes creation of complex information products as one of its goals. We are especially motivated by their potential to change the practice of software engineering, which is fundamentally about the principles, methods, and processes involved in producing complex information products. The Web could enable these products and their associated processes to be dispersed globally, while remaining highly interconnected and dynamic. Software products could be better designed and constructed, since globally dispersed teams of specialists—designers, analysts, programmers, testers, and others—could be assembled in cyberspace. Chosen to fit the particular needs of a development task, such teams could be assembled, even for short-term collaboration. Enabling their products to stay linked to their development environment would improve the quality of both the software and its support organizations. Moreover, such links could support optimization based on observed usage patterns, in-field updates, and ongoing quality assessment. Training in product use and system maintenance and evolution could be facilitated through links to process support in the development environment. Software reuse could also be greatly increased, as a convenient worldwide marketplace of software components develops.

While the Web has already proven its value in many application domains and situations, the changes we recommend will help yield an infrastructure rich enough to support creation and maintenance of complex information products and thereby the flourishing of virtual enterprises. C

References

1. Anderson, K. Integrating open hypermedia systems with the World-Wide Web. In *Proceedings of the 8th ACM Conference on Hypertext* (Southampton, England, Apr. 6–11). ACM Press, New York, 1997, pp. 157–166.
2. Berners-Lee, T. WWW: Past, present, and future. *Computer 29,* 10 (Oct. 1996), 69–77.
3. Carr, L., Hill, G., De Roure, D., Hall, W., and Davis, H. Open information services. *Comput. Networks ISDN Syst. 28,* 7–11 (May 1996), 1,027–1,036.
4. Cutkosky, M., Tenenbaum, J., and Glicksman, J. Madefast: Collaborative engineering over the Internet. *Commun. ACM 39,* 9 (Sept. 1996), 78–87.
5. Fielding, R. Maintaining distributed hypertext infostructures: Welcome to MOMspider's web. *Comput. Networks ISDN Syst. 27,* 2 (Nov. 1994), 193–204.
6. Fielding, R., and Kaiser, G. The Apache HTTP server project. *IEEE Internet Comput. 1,* 4 (July–Aug. 1997), 88–90.
7. Fielding, R., Gettys, J., Mogul, J., Nielsen, H., and Berners-Lee, T. Hypertext Transfer Protocol — HTTP/1.1. Internet Proposed Standard RFC 2068, Univ. of California, Irvine, DEC, MIT/LCS, Jan. 1997.
8. Gronbaek, K., and Trigg, R. Design issues for a Dexter-based hypermedia system. *Commun. ACM 37,* 2 (Feb. 1994), 41–49.
9. Kaiser, G., Dossick, S., Jiang, W., and Yang, J. An architecture for WWW-based hypercode environments. In *Proceedings of the 19th International Conference on Software Engineering* (Boston, May 17–23). ACM Press, New York, 1997, pp. 3–13.
10. Malone, T., and Crowston, K. What is coordination theory and how can it help design cooperative work systems? In *Proceedings of the Conference in Computer-Supported Cooperative Work* (Los Angeles, Oct. 7–10). ACM Press, New York, 1990, pp. 357–370.
11. Maurer, H. *HyperWave: The Next-Generation Web Solution*. Addison-Wesley, Harlow, England, 1996.
12. Slein, J., Vitali, F., Whitehead, E., and Durand, D. Requirements for a distributed authoring and versioning protocol for the World-Wide Web. Internet Informational RFC 2291, Xerox Corp., Univ. of Bologna, Univ. of California, Irvine, Boston Univ., 1998.

Roy T. Fielding (fielding@ics.uci.edu) is a Ph.D. student in information and computer science at the University of California, Irvine, a coauthor of the proposed Internet standards for HTTP and URL, and a cofounder of the Apache project.

E. James Whitehead, Jr. (ejw@ics.uci.edu) is a Ph.D. student at the University of California, Irvine, and chair of the Web distributed authoring and versioning working group of the Internet Engineering Task Force.

Kenneth M. Anderson (kanderso@ics.uci.edu) is a member of the research staff in the Department of Information and Computer Science at the University of California, Irvine, and the designer and developer of the Chimera open hypermedia system.

Gregory A. Bolcer (gbolcer@ics.uci.edu) is a Ph.D. student in information and computer science at the University of California, Irvine, and founder of Endeavors Technology, Inc.

Peyman Oreizy (peymano@ics.uci.edu) is a Ph.D. student in information and computer science at the University of California, Irvine.

Richard N. Taylor (taylor@ics.uci.edu) is a professor of information and computer science and director of the Irvine Research Unit in Software at the University of California, Irvine.