Dah Ming Chiu
David M. Griffin

Building Collaboration Software for the Internet

Collaboration software for the Internet's World Wide Web involves the development of shared information systems for network computing. The AltaVista Forum version 2.0 software from Digital contains extensions to World Wide Web technology that facilitate collaboration on the Internet. The extensions consist of a toolkit and a set of collaboration applications. The toolkit components include a built-in database with an indexing and search capability. Generic applications include discussion, document sharing, and calendar applications and administrative functions for managing users, teams, and access control.

The Internet and the World Wide Web (WWW) have changed the scope of network computing. As the Internet user population has grown, so has the demand for better ways to collaborate on the Internet. Some examples include the ability to share and discuss issues of common interest, coauthor documents, and track project status. Although today's WWW is ideal for publishing information, it requires considerable customized programming to support collaboration. The AltaVista Forum version 2.0 product is both a set of collaborative applications and a toolkit (platform) that facilitates easy, efficient, and rapid development of collaborative applications for the Internet for both UNIX and Windows NT systems.

In this paper, we describe our experiences in building collaboration software for the Internet. We begin with a brief discussion of WWW technology and groupware applications. Then we present our design philosophy and the framework of the software and discuss the applications supplied by AltaVista Forum. Following that, we discuss the various experiences gained in developing software for the new Internet paradigm. We conclude the paper by discussing our plans for future development efforts.

World Wide Web Technology

Today's Internet was originally a government-funded computer network that facilitated collaboration among academic researchers. Information exchange was conducted by means of electronic mail (e-mail) and file transfer. Over time, bulletin-board style discussions were supported by the Network News Transfer Protocol (NNTP), which propagated textual discussion threads to a large number of NNTP servers for viewing. With the development of the WWW technology, collaborating over the Internet has become even easier.

The WWW technology consists of the following elements:

- Universal resource locator (URL), a convention for information naming and linking
- Hypertext markup language (HTML), a text-based language for information rendering

- Hypertext Transfer Protocol (HTTP), a simple client-server protocol to transport information associated with a URL
- Web browser, a program that renders HTML documents, provides URL caching, and supports a directory for URLs
- Web server, a server that responds to requests for information from the Web browsers

Information Access

WWW technology has transformed the way users access information through computer networks. Access to information on the Internet was primarily text-based; with the WWW, users are able to access information in multimedia format. The combination of functionality (information linking, graphical interface, and caching), extensibility (for dealing with new protocols and new information types), ease-of-use, and low cost appealed to a wide range of users in homes, offices, and corporations. In addition, the Mosaic-style of "point-and-click" graphical Internet browser has become the most widely accepted user interface for network computing.

The most popular use of the WWW today is for publishing information, and the process is comparable to the way a newspaper publishes or a television station broadcasts information. The roles of the information provider and the information consumer are clearly defined. The information provider gathers and organizes the pertinent information, converts it to the HTML scripting format, and makes it available on a Web server. The information consumer, after obtaining initial access to the Web server (as one might tune into the correct television station), can then browse and search for various types of information available on that server. The linking capability of URL and HTML allows the references or links to additional information on various servers to be easily published along with the original information.

In contrast, multiple information providers work in collaboration to generate the content of shared information. For the purposes of this paper, we will assume that there is only one type of user—information collaborators.

Collaboration and Groupware

The WWW is useful for many types of collaboration. For example, a project team may need to keep track of project status and individual progress; people with a common interest (e.g., film enthusiasts) may want to share and discuss their views on that topic; a customer support group may need a system to provide on-line answers to real-world customer problems; or several authors may wish to work on a document together.

Today, several computer applications facilitate such collaboration. Collectively, these applications are known as groupware. Lotus Notes is a popular groupware application. Typically, groupware applications support the following capabilities:

- Management of a set of users and groups
- Storage of shared information in a database (sometimes with replication capability)
- Viewing information stored in the databases by means of a graphical interface
- Protection of the collaboration environment when necessary through authentication and access control

Groupware systems are built to run in homogeneous client environments, such as the Microsoft Windows environment. They rely on specific client-server technology, which is often proprietary, to support remote operations.

The popularity and rapid growth of the Internet and the WWW have created an open, universal, and easy-to-program infrastructure that can readily serve several groupware functions. Engineers at Digital's Internet Software Business Group recognized the potential of using the WWW as the underlying infrastructure for groupware solutions and at the same time saw that the groupware applications available today have features that the WWW lacks. Our goal was to add groupware features to the WWW to facilitate collaboration.

We started exploring the idea of using the Internet and the WWW for groupware applications in the summer of 1994. By the end of that year, we had built a prototype that supported the simplified discussion (bulletin-board) features of an internal product known as DEC Notes.[1] This prototype generated considerable interest among active DEC Notes users who were seeking a similar solution built around an Internet infrastructure. Based on their feedback, the prototype was redesigned and became a product.[2]

By September 1995, we had built several collaborative applications to run over the WWW. In a workshop organized by the World Wide Web Consortium and the Massachusetts Institute of Technology, we participated in discussions on how to extend the WWW technology to support collaboration. All the workshop participants presented their ideas to the WWW Consortium for review.[3]

Design

In this section, we summarize our design philosophy and discuss the framework and applications developed for the AltaVista Forum product. For our design, we adopted an object-oriented approach, which meant that we would have to modularize the various components for reuse and modification.

Design Philosophy

Our fundamental design philosophy required using the Internet and its infrastructure as building blocks for our collaboration software. After years of experimenting and collaborating to develop an open process, the Internet developers realized that the Internet had reached a state of critical mass. In the case of networks and connectivity, reaching critical mass is a tremendous impetus for agreeing on a common standard. As more and more users access the Internet, the need for software development for the Internet also increases. In addition, the very nature of the Internet demands an open standardization process to ensure the long-term viability of a product.

Our philosophy also included the reuse of existing open software as building blocks whenever possible. In addition to our choice of building upon the Internet and the WWW technology, we selected the Tool Command Language (Tcl) as the primary language for developing most of our application and user interface functions.[4] We also took advantage of the database library in the Berkeley UNIX distribution for built-in database support.[5]

Another objective was to make sure our software would be easy to port to all the relevant operating system platforms. This principle guided our selection of components and helped us isolate a small set of platform-dependent functions into a special library for porting the software.

As stated earlier, we tried to take an object-oriented approach whenever possible. The advantages of our approach became increasingly apparent as more people became involved with the software development. The object-oriented approach made component reuse feasible.

Framework

Our framework organizes the AltaVista Forum software into two layers: toolkit and applications. The tools required to build the applications overlap each other. We have used them to build generic applications, including a discussion application that supports users discussing a set of related topics, much like newsgroups do; a calendar application that supports users' abilities to schedule events on a specific date and at a particular time; and a newspaper application that provides a personalized news filtering service. We envision that, over time, the framework we have developed will support a number of diverse applications. Figure 1 shows the AltaVista Forum toolkit and application layers.

The toolkit is a combination of both C and Tcl code that creates the following interface components:

- Built-in database. The application uses a built-in database to store its object instances. The database is a very simple relational model with an object hierarchy relationship facility available to those applications that need it. The library also provides inversions on certain attributes to support fast retrieval and sorting based on attribute values.
- Built-in indexing and search. An indexing and search function complements the database by providing a high-speed query facility. For less-structured objects, it is often easier to index them and look them up using a search tool.
- Graphical user interface support. The use of a graphical user interface insulates applications from having to deal with HTML directly and cope with its changes over time. Abstract definitions of user interface objects also tend to simplify and clarify the code and create a more uniform appearance on the screen.
- Access control. All applications require some form of access control to regulate who can access, create, modify, and delete various objects.
- Internationalization. An internationalization facility gathers strings that appear in the user interface into message catalogs for later translation to different languages.
- Platform-specific support. A special library isolates those operating system—dependent functions that vary from platform to platform. Certain file system accesses and date/time library accesses are examples of this component.

Armed with all the components in the toolkit, an AltaVista Forum application consists of a set of functions, each responding to a different user request. The organization of an application is modular. A function can call various objects that are defined separately as part of the application, including the following:

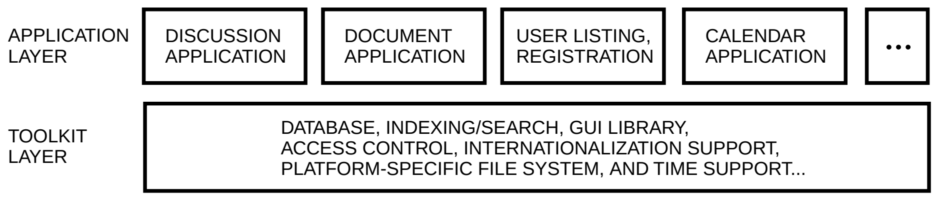
APPLICATION LAYER
DISCUSSION APPLICATION
DOCUMENT APPLICATION
USER LISTING, REGISTRATION
CALENDAR APPLICATION
...
TOOLKIT LAYER
DATABASE, INDEXING/SEARCH, GUI LIBRARY, ACCESS CONTROL, INTERNATIONALIZATION SUPPORT, PLATFORM-SPECIFIC FILE SYSTEM, AND TIME SUPPORT...

Figure 1
AltaVista Forum Toolkit and Application Layers

- Graphical objects such as definitions of buttons, toolbars, various objects that are part of a form (e.g., select boxes, radio buttons, check boxes, text boxes), and icons.
- Database entries, the definitions of their attributes, and default values.
- User interface aggregate objects such as forms, views, dialogs, and error messages.
- Default access control policies, including default groups, access rights, and their mappings, to control who can access individual forums and what actions they can take within them.

This approach encapsulates the details in low-level modules, making the software more readable and maintainable. It also makes it easy for different functions to reuse the objects.

To further facilitate code sharing, the framework also allows applications to inherit a set of functions and objects that have been grouped together as a pseudoapplication. For example, the access control management functions can be grouped into a pseudoapplication and certain button and toolbar definitions can be grouped into another pseudoapplication. All applications that need access control and the common graphical objects that lend a consistent "look-and-feel" can inherit those functions and objects from pseudoapplications.

The AltaVista Forum product works in conjunction with the Web browser and the Web server. The Web browser submits requests to the Web server whenever the user opens a link. If the link points to a file, then the Web server sends the file to the browser, which is the normal interaction. The link can also point to programs on the server; in this case, the Web server invokes the program and then the program responds to the user.

When the link points to the AltaVista Forum, the Web server invokes the AltaVista Forum dispatcher program through the common gateway interface (CGI). Based on the information passed along with the user request, the dispatcher invokes a specific application, which, in turn, calls various tools in the toolkit to respond to the user's request. Figure 2 illustrates the interaction of the AltaVista Forum software with the Web browser and server.

Parameters are passed to the dispatcher from segments of the URL. The dispatcher parses the URL into the pieces that provide the overall control of the program: (1) the forum name, (2) the access control area name, (3) the message name, and (4) the message arguments.

Each forum is an instance of an application object. For example, many discussion forums are available on various topics. Each discussion forum has its own name at the time of creation; however, the same discussion application can be used to manage all the forums.

An access control area contains a set of forums and a common user/group database. An administrator group helps administer the user/group database and establish overall access control policies for the environment. A user registers only once with an access control area. Based on the access control area location, the hypertext server not only knows where to find the user's credentials for authentication purposes but also knows how to authenticate the user and pass the authenticated user identity to the AltaVista Forum environment. Given the user identity and the access control location, AltaVista Forum software can also look up the user profile, check access control, and perform other user-specific functions.

The message name and message arguments then select particular actions to perform within the application.

Generic Applications

The AltaVista Forum product supplies a set of generic applications that make the software immediately usable. The applications are described in this section.

User and Group Management and Lookup This application provides an interface for user registration (either by the user or by an administrator). Users can supply and modify their business card information such as phone numbers and e-mail addresses. Users can also set certain preference parameters that help the AltaVista Forum software tailor its responses (e.g., native language and preferred display formats). In addition, groups can be created and modified as a set of users. This application also provides the interface for listing and searching for user and group information for all forums. As discussed earlier, the AltaVista

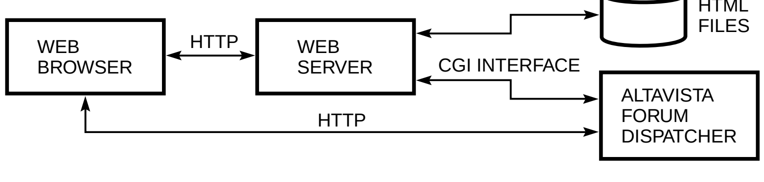

Figure 2
Interaction of AltaVista Forum, Web Browser, and Web Server

Forum product can support all the users that a Web server can handle since only one repository of users and groups is necessary.

Community, Team, and Personal Vistas A vista is another term for home page, which is a place for the user to log in to the WWW. Once in the community vista, the user sees a set of public forums and links to perform various tasks, e.g., register oneself, look up teams or join a team, perform AltaVista Forum administrative tasks (if an administrator), and so on. For this reason, the community vista is also called the summit. In much the same way, a team vista keeps track of all the forums and links for a group of users, and a personal vista performs this function for a single user. Both team and personal vistas can own forums that are not visible to the public community vista.

Discussion Much like a bulletin-board discussion group or Digital's DEC Notes software, this application permits users to share ideas on a set of related topics. Users create topics and replies that form a hierarchical tree (also known as threaded topics), providing a way for users to navigate through existing discussions. Other methods of reading the existing discussion are also provided. These include chronologically navigating through items not read; listing unread items only and selectively reading them; and searching for topics and replies containing certain words that were entered during a particular time period by a certain author. Users can also create multiple discussion forums to discuss different topics; this is true for the following applications as well.

Document Sharing The document sharing application enables users to organize documents of the same type into hierarchically organized folders. In addition, it keeps track of versions of the documents, attachments, and comments. As with the discussion application, users can browse through and search for specific documents using a variety of methods.

Newspaper The newspaper application lets users select a specific source of information and then define filters to present only those items of potential interest. A good example of an information source on the Internet is one of the real-time news feeds. Using the newspaper application, it is also possible to read and monitor other information sources, e.g., e-mail sent to a distribution list or information appearing on a set of WWW sites.

Calendar The calendar application permits users to enter a set of scheduled events (or a to-do list) and present the events as a calendar (sometimes called a diary). The application supports requests to add items to the calendar, thus allowing the calendar to be used as a scheduling tool. Although a calendar forum can be set up for each person, it is equally useful to have a team calendar, a community calendar, or even a calendar for a specific type of event.

Experiences

In this section, we summarize some of our experiences and discuss the lessons we learned along the way. As a result of our decision to rely on the Web browser as the universal user interface, we had to resolve some unique user interface issues. Because we chose to use Tcl for developing higher-level objects, we had to cope with using an interpretive language. We designed the database and indexing and search interfaces based on extensibility and portability goals. Finally, in the design of access control, we had to carefully weigh the pros and cons of simplicity and flexibility.

Coping with the User Interface Defined in HTML

Very early in the design phase, we decided to make AltaVista Forum client-independent, with the exception of dependence on the Web browser. This decision was based on the fact that the Web browser was already freely available on most of the platforms. We expected the browser to become a ubiquitous network front end, allowing us to focus on building groupware functions on the server. This meant that we were faced with the task of designing the user interface using HTML.[6]

Since HTML was evolving, our first step was to define graphical objects in more abstract constructs supported by our toolkit. Each construct encapsulates the specifics into a representation of a graphical artifact in HTML in the toolkit. Thus as HTML evolves, or as the page design changes, only one area needs to be updated. For example, a select box object on a form may be defined as follows:

```
forum selectbox s.language \
     -mapto language \
     -label "Select a language:" \
     -labelbreak

s.language add_option English 1 selected
s.language add_option French 2
```

In this example, a select box is defined to begin with a label and some spacing and then to contain two options: English and French, with English as the default. The values 1 and 2 are internal representations of the selected values. Also, the "map-to" switch specifies that this object must correspond to the language attribute in the database, a feature that was included to simplify database update.

Note that although a label is specified, no specification is provided to represent that label in a particular font or typeface. Neither is the actual spacing for label break specified. These decisions are made in the forum select box part of the toolkit procedure, which translates this object into HTML.

Most of the early Web browsers were single-window based. This limitation was especially problematic for us because most of our applications provide some organization to the information content. A much more natural way of browsing for our environment would include at least two windows: one showing the context and the other showing the content of a specific item. For this reason, we introduced multiple navigational methods. For example, the discussion application

- Allows hierarchical navigation (previous, next, up)
- Allows navigation in chronological order (next unseen, what's new)
- Provides a category view that lists topics according to their category
- Supports content-based search or an index-like function

Newer versions of Web browsers support frames, which have multiple window-browsing capabilities (although the standards in this area are still a bit vague). We are updating our applications to take advantage of these new features.

Usability studies guided our decisions as we were designing forms and dialog boxes. It is likely that many potential users of our product are familiar with Windows-style user interface objects. Because the early Web browsers (e.g., Mosaic) were UNIX-based, little attention was given to providing a human–computer interface that resembled the more widely used Windows interface. However, our usability studies indicated that many personal computer (PC) users had difficulty using Web browsers out-of-the-box. For example, a user might expect a dialog box to have certain standard buttons, such as OK, cancel, and clear. Ideally, the user would know what to do with these buttons without any training. To make our software easy to learn, we tried to follow the same user interface style that was already familiar to most users. Since we were limited by HTML and browser design, this was not a simple task. Thus we were often forced to produce rough facsimiles of the more well-known interface artifacts.

In summary, we found usability studies to be extremely valuable when designing end-user applications. For this reason, it is important to allocate enough time in the product design cycle to collect user feedback before beginning product development.

The Pros and Cons of Using an Interpretive Language

As mentioned earlier, we selected Tcl as the language for building the AltaVista Forum toolkit. Tcl is a highly portable, extensible, and freely available language that was originally designed to be embedded in a larger framework.[4] However, it is also an interpretive language, which supported our goal of rapid and iterative development of collaborative applications for the WWW.

We extended standard Tcl to provide a set of commands and objects that formed the AltaVista Forum toolkit: database, HTML generation, access control, internationalization, user profile management, and platform-specific support. Many of these extensions supported an object-based environment (i.e., the environment supported standard Tcl objects and our simple inheritance mechanism). The use of these extensions made it easier to develop applications than it would have been with Tcl (or any other language), alone. As a result, these extensions form the basis for future development tools.

From the beginning, we knew that the choice of an interpretive language was going to involve trade-offs. In fact, performance, which was our most critical trade-off, continues to be a concern for the engineering team. Although the performance of an interpreted language is lower than that of a compiled language, fast processors have made the use of an interpreter worthwhile because of the reduced expense of developing applications. The use of Tcl in the AltaVista Forum software certainly takes advantage of this. Although the applications and part of the toolkit are written in Tcl, many critical parts are implemented in a compiled language (such as C) to stay within performance requirements. The engineering team is continually searching for ways to improve performance while accommodating requests for new features and tracking the rapidly evolving WWW environment.

The second trade-off was the absence of a sophisticated debugging and profiling environment. Partly due to the limitations of Tcl and partly due to the stateless nature of WWW transactions, some of the more sophisticated development tools that programmers expect to see are not readily available. Despite these shortcomings, rapid development is still possible; however, we expect even larger gains as we correct these problems in the future.

Interfacing to the Database

Several factors (primarily portability and cost) influenced our decision to build a hybrid database rather than the more customary relational database. The database in the AltaVista Forum toolkit consists of a B-tree indexed file (from the Berkeley ndbm package) for storage of basic attributes about documents, which is backed by the file system for the nonstructured data. This design, combined with the search engine (described in the next section), is quite effective for the types of applications we initially developed with the AltaVista Forum toolkit.

In effect, the database is organized as a collection of documents (or entries) that have unique identifiers (document IDs), hierarchical document numbers, and a set of attributes that is similar to a relational database table. The toolkit provides each entry with a set of built-in attributes (such as title, creation and modification dates, and author). The applications can then deliver additional attributes.

The toolkit provides the means to retrieve, modify, and iterate through the collection of entries in a straightforward manner. Because the attributes are part of the application description and are not stored in a separate database, the toolkit can use its knowledge of the attributes to simplify certain common operations. For example, because transferring data from HTML forms to the database and back is a basic operation in collaborative applications, the toolkit can link fields on forms to database attributes, making it possible to store them with a single command. To support a dynamic development environment, the toolkit also upgrades databases in real time as new attributes are added or deleted. This permits the application developer to concentrate on the task at hand rather than worry about database management tasks.

Although the primary organization mechanism is a flat table indexed by document identifiers, the database integrates a hierarchical relationship between entries when necessary. Because hierarchies are common in collaborative applications (e.g., folders/documents and topics/replies), it was important to reflect this in a natural way in the database.

In addition to attributes, the database offers properties. Compared to attributes, which are stored for each entry in the database, properties are stored within each forum. Application designers can use these properties in any way they desire: they are simple key-value relationships. The AltaVista Forum software uses properties to implement a variety of features, from access control policies to the background color of the screen display.

User properties are an extension of standard forum properties. They act like forum properties except that they are tied to the user who is executing the transaction. User properties keep database locking to a minimum because, in collaborative applications, a user will typically execute only one transaction at a time.

Indexing and Search: The Way of the Future?

One key design decision was to include an indexing and search engine as a basic component of the product. Although the database is often the central piece of a groupware product, an indexing and search engine often plays a similar role for a WWW site. This development is completely consistent with the philosophy of the WWW—information is linked as needed, not necessarily following any structure. Database use is more suitable for information objects that have some uniformity in their definitions.

The basic function of the indexing engine is to map a set of words to a document containing those words. (The term document is used in a generic sense. It can be any logical entity associated with a set or words.) The indexing information must be stored in such a way that subsequent searches based on individual words (and phrases) are efficient and speedy. The indexing engine in the AltaVista Forum toolkit is basically the same indexing engine available on the AltaVista Web site.[7] Designed and implemented at Digital's System Research Center, it is highly scalable and efficient.

The built-in database functions as a repository for entries with a predefined set of attributes. It provides fast retrieval when the entries are identified using either an entry ID or a hierarchical ID, and it provides simple creating, updating, and sorting functions associated with retrieval. The indexing and search engine complements the AltaVista Forum database: it provides a content-based search method and functions at higher speed. Since the search engine is extremely fast and scalable, we also use it to index some of the attribute values in the database. This allows us to use the search engine for certain compute-intensive searches that otherwise would be performed by the database.

Based on our experience, we expect the capabilities of the indexing and search engine to continue to expand. As the popularity of the WWW technology continues to grow, the volume of published information will also increase. Only a small amount of this information can be effectively captured in databases. The indexing and search engine is an invaluable tool for mining useful information out of the vast amount of data stored in these databases.

The Dilemma of Access Control

Designing access control is very challenging because users and administrators have different requirements. On the one hand, administrators want a high degree of flexibility in controlling access. Their issues include the following:

- What type of information is subject to access control?
- Should access control be defined for every possible access/action type?
- Should there be arbitrary flexibility in defining groups (including nesting)?

On the other hand, users have stated that they do not like products in which access control operations are complex, especially in the case of a product that is supposed to help people collaborate. In a majority of scenarios, they argue that very little access control is needed.

For this reason, we tried to strike a balance between administrators' needs and users' preferences. Although we recognize the importance of access control, we did not give it precedence over product usability. Since usability was our priority, and the time available to work on it was limited, we divided our efforts between making access control flexible and choosing default options that would promote collaboration.

We defined access control for the whole database (forum), rather than for individual entries and attributes of entries. However, some entry-level access control is necessary. For example, it is preferable to let only the owner (or the creator) of an entry modify and delete that entry. As a result, we allowed the group definition to include entry-specific logical users, rather than provide a general mechanism for entry-level access control. Therefore, a group may contain a member who is the owner of the current entry. During access control checking, the current entry's owner is looked up and matched against the currently logged-in user.

Instead of letting the administrator define access control for each possible incoming access/action, our framework allows the application definition to group accesses together into logical access rights. For example, for the discussion application, we defined the following access rights:

- Read—Includes all read URLs (different views, whether for a single entry or a list of entries)
- Contribute—Includes adding a topic or reply
- Modify—Includes any form of modification or deletion
- Moderate—Includes such functions as creating keywords, polling options, controlling number of levels of replies, and setting certain entries as hidden
- Administrate—Change access control or other kinds of resource consumption policies

By defining these access rights, the administrator only needs to establish who can do these five operations, rather than define numerous other kinds of operations. It is still possible to change and add to this group of access rights by making simple modifications to the application definition.

Our basic strategy for making access control easy to manage is to set up default policies of access control that apply to as many situations as possible, within reason. The default policy is added to the application definition. If the administrator is satisfied with the default policies, then the access control can be used as supplied. For the discussion application, the default policy is the following:

- Read—All users, including anonymous
- Contribute—All users, excluding anonymous
- Modify—Owner (creator) of entry and moderators
- Moderate—Owner of the forum
- Administrate—Owner of the forum

To simplify implementation, we chose not to allow nesting of groups. Our design allows for adding it in the future as long as it makes management of access control policies easier.

Future Directions

To date, we have received encouraging feedback from users. Of the ways that we can continue to improve the AltaVista Forum product, we feel the following deserve the highest priority.

First, we need to provide better ways to help users deal with information overflow. Although we have built ways to filter and search information into our application, further simplification is necessary. We are working on smart agents that bring the relevant information to the user's fingertips.

Second, a number of the functions that we provide can be more easily performed on the client machine. The Java language is the best candidate for providing these functions since it enables us to handle a wide variety of client platforms. Initially, we are looking into using Java to improve certain user interface problems, such as opening additional windows on the client machine to notify users of new information.

Third, synchronous collaboration using video, audio, and whiteboard will soon become feasible and cost effective. It is important for us to help bring users together through both synchronous and asynchronous methods of collaboration. For example, users should be able to use the calendar application to schedule a meeting over the Internet, and Windows should be available to the user automatically.

Fourth, as the AltaVista Forum software matures, we hope to add to its performance and increase its scalability. As its environment evolves, we are looking into ways to bypass the CGI interface and use a compiled language for more of the toolkit implementation. We also hope to add support for large commercial databases.

Finally, we will continue to add innovative applications to our product. We recently built a prototype of a customer-support application that keeps track of problem reporting. We are looking into other applications such as project management, group review, and survey and decision-support systems.

Acknowledgments

We wish to thank the AltaVista Forum development and management teams for their contributions to the product. In particular, we wish to thank Peter Hurley for his leadership in starting the effort; Ralph DeMent, Bob Travis, David Marques, and Rick Frankosky, who have worked with us throughout the lifetime of the product and with whom we have developed a special camaraderie; and Dan Kalikow, who was the first adopter and has cheered us on ever since.

References and Notes

1. DEC Notes is a discussion application running primarily on VAX systems connected on a DECnet network. Still a very popular tool within Digital, it is used for collaborating on many topics, ranging from product development, customer support, and marketing to various personal interest topics.
2. The product was originally called Workgroup Web Forum. It was subsequently merged into a larger family of products and the product name became AltaVista Forum.
3. For more information on the World Wide Web Consortium/MIT Laboratory for Computer Science Workshop on the World Wide Web and Collaboration held September 11–12, 1995, see http://www.w3.org/pub/WWW/Collaboration.
4. J. Ousterhout, *Tcl and the Tk Toolkit* (Reading, Mass.: Addison-Wesley Publishing Company, 1994).
5. *NDBM(3), 4.3 BSD Unix Programming Manual Reference Guide* (University of California, Berkeley, 1986).
6. During this time, Java was still on the drawing board, or at least not generally supported by Web browsers. We did expect to use Java to enhance our user interface over time.
7. For access to Digital's indexing and search engine, visit the AltaVista Web site at http://altavista.software.digital.com.

Biographies

Dah Ming Chiu
Dah Ming Chiu was a consulting engineer in Digital's Internet Software Business Unit and a technical leader in developing the AltaVista Forum groupware product for the Internet. Before that project, he worked for the Networks Architecture Group on congestion and flow control, network monitoring, name service, and the X.500 standard. Previous to that, he worked on performance modeling and analysis network protocols and graphical workstation design. Dah Ming is currently an architect in the Internet Solutions Group of Sun Microsystems, Inc. He received a Ph.D. in applied mathematics (1980) from Harvard University and a B.Sc. in electrical engineering (1975) from the Imperial College, London University. He holds three patents in the areas of congestion control and network monitoring and is a coauthor of *Network Monitoring Explained.*

David M. Griffin
Dave Griffin joined Digital in 1981. He is a principal software engineer in the AltaVista Collaboration Engineering Group, where he leads the AltaVista Forum Toolkit team for version 3.0. Dave also led the toolkit team for version 2.0 and was the primary designer and implementer of the (Workgroup Web Forum) version 1.0 toolkit and the author of the document-sharing application for version 1.0. Prior to this work, Dave led the DECdns server project (part of the DECnet/OSI program) and designed and implemented the hierarchical cells and cell-renaming facilities in the DCE Cell Directory Service. He has been involved in the development of a number of distributed information systems for Digital and other companies. He holds two patents in distributed systems technology.