The EXPRESS Web Server: A User Interface for Standards Development

David A. Sauder
National Institute of Standards and Technology
100 Bureau Dr., Stop 8260
Gaithersburg, MD 20899-8260 USA
(301) 975-3536
david.sauder@nist.gov

Joshua Lubell
National Institute of Standards and Technology
100 Bureau Dr., Stop 8260
Gaithersburg, MD 20899-8260 USA
(301) 975-3563
lubell@nist.gov

1. ABSTRACT

The EXPRESS Web Server is a World Wide Web interface for standards developers creating specifications for STEP, the STandard for the Exchange of Product model data (officially ISO 10303). The Server enables users to run applications needed to build and populate EXPRESS specifications using a web browser and without having to install and configure the applications locally. The Server also provides standards development teams with an infrastructure for collaboration. This paper describes the Server's user interface, specifies how the Server's architecture addresses several design challenges, and discusses the impact newly emerging web technologies might have on future enhancements.

1.1 Keywords

STEP (ISO 10303), EXPRESS, EXPRESS Web Server, CGI, Tcl, cgi.tcl, HTML, WWW, dynamic web pages

2. BACKGROUND

STEP (the STandard for the Exchange of Product model data, officially ISO 10303) [6], is a family of standards specifying a description of product data throughout the product's life cycle in a computing-platform-independent manner. Each individual standard in STEP defines the information requirements for a particular area of design, manufacturing, engineering, or product support. STEP provides a neutral representation for sharing product data among dissimilar software applications. STEP standards are developed using a set of reusable constructs that serve as building blocks. The standards and building blocks are specified using EXPRESS (ISO 10303-11) [7], an information modeling language intended to be interpretable by computers as well as by humans.

Although several STEP standards have been completed to date and a couple of dozen more are in the pipeline, the standards development process is slow and expensive. Indeed it typically takes a team of four to six people more than a year just to create an initial draft specification. Two reasons why development of STEP standards are so cumbersome are:

- Teams are geographically and organizationally dispersed, making collaboration between team members and configuration management of EXPRESS specifications a challenge.
- Developers often lack the tools[1] they need to build EXPRESS specifications and populate them with data. The reason for this is that, although the potential STEP end-user community is quite large, the STEP standards developer community is too small to make it affordable for vendors to provide low-cost software applications tailored for standards development. Because standards development is a voluntary activity, budgets tend to be tight, making it hard for teams to spend large sums of money on software.

In order to address these concerns, the National Institute of Standards and Technology (NIST) maintains a service which enables developers of STEP to run EXPRESS parsers and other necessary applications remotely without having to install or maintain them locally. Users with only a web browser can run applications that may otherwise not be available for use on their computer platform. Additionally, the need for EXPRESS tool providers to port their applications to other platforms is lessened. The service also allows users to upload and download EXPRESS specifications and data sets, share them with other team members, and access common EXPRESS definitions from a repository of building blocks. Two recent enhancements to the service are:

- A World Wide Web interface. Because the initial implementation predated the popularization of the web, interaction between users and applications took place using an email interface similar to that of a mailing list server. The new web interface allows for greater interactivity and makes the service easier to use.

[1] Names of companies and products are provided in order to adequately specify procedures and equipment used. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the products are necessarily the best available for the purpose.

- Additional applications. The original implementation [12] provided access to a set of EXPRESS applications all developed by NIST using the same software tool kits. Today, however, there is a wider variety of EXPRESS tools available. Therefore, access to additional tools has been provided.

The remainder of this paper describes the newly enhanced "EXPRESS Web Server," emphasizing topics and issues specific to the web. First, an overview of the Server's design and functionality is provided. Next some challenges in designing the Server are enumerated, and the strategies adopted to meet those challenges are described. Finally, roles the emerging web technologies might play in future versions of the Server are discussed.

3. EXPRESS WEB SERVER DESIGN AND FUNCTIONALITY

The Server provides users with:

- read-only access to a repository of common standardized EXPRESS "building blocks", enabling users to build EXPRESS specifications referring to objects defined in the common building blocks.
- read/write access to a private area with access privileges granted upon providing a user ID and optional password, enabling users to store their own personal specifications and data files.
- a uniform user interface for running the various applications supported, hiding the idiosyncrasies and user interface differences of individual tools from users.

Access to the private area not only allows users to read and write their own files, but also allows them to share their files with anyone else who knows their web-site specific ID and password. Thus, a team developing a STEP standard can build and populate their EXPRESS specification collaboratively by requesting a single ID and password and granting all team members access to the team's private area on the Server.

The web pages comprising the Server's user interface are generated on demand using CGI (the Common Gateway Interface) [2] and were implemented in Tcl (Tool Command Language) [17] with the cgi.tcl [13] CGI support library. CGI scripts are used to create these pages, keeping the maximum number of HTML (HyperText Markup Language) [18] pages needed to fulfill a user's request down to three. Each of these three pages are created dynamically using a CGI script. Here is a content summary of the three pages:

- The first page requests an optional user ID and password and provides a selectable list of all the server commands available.
- The second page either displays the command results or requests the input needed for use with the command selected on the first page.
- The third page (if needed) displays the results of the command.

Where appropriate, form input from the second or third page drives a CGI script which is dedicated to downloading or viewing a file. When a download action is performed, the script generates output causing the browser's download window to appear for downloading the script output.

The first page serves as an entry point for users. It contains a form, shown in Figure 1, for entering an ID number with optional password and choosing from a pull-down list of command options. A cookie [9] is used to save the ID value entered. The most-recently-entered user ID is always saved to the user's local cookie file. Each time the first page of the Server is loaded, the ID field is initialized to whatever value

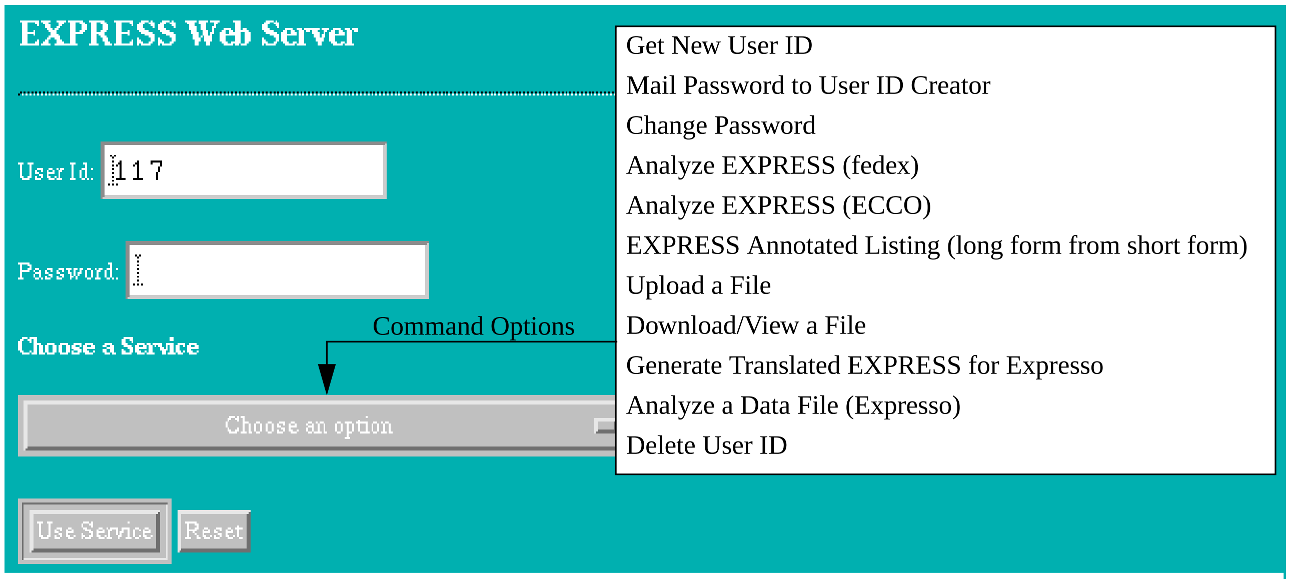


Figure 1. The EXPRESS Web Server Start-up Page Where a User Optionally Enters a User ID (and Password) and Selects a Command to be Executed

is stored in the user's cookie file.

The second web page is generated using the form input from the first page. If the command selected on the first page requires no user-directed input (e.g. "Delete user ID") then the command is executed and the results are incorporated into the second page's page content. If the command requires user-directed input values, the second page dynamically creates a page that requests all the necessary user input for executing the selected command. The second page requires dynamic page generation since input to many of the commands requires that the user select a file from a directory on the Server. The files on the Server are added and deleted dynamically by the user and therefore cannot be hard-coded as static HTML.

Figure 2 shows the second page's appearance if a user selects "Download/View a File" from the pull-down list on the first page. Using the form on this page, a user specifies the file to be obtained and whether the file's contents should be downloaded or displayed in the browser. The file chosen may either be a file stored under the user's ID or one of the public "building block" files. User files are separated into several categories: e.g. EXPRESS specification files, files containing data sets, and files containing output saved from Server commands previously executed. The public EXPRESS files are grouped according to whether they are from the initial versions or revised versions of STEP standards.

Figure 3 shows what the second page looks like if the user chooses to check the syntax of an EXPRESS specification using "fedex" [10], one of the parsers available through the Server. The form on this page allows the user to choose whether to wait for fedex to generate diagnostics or to run fedex in the background. If the user chooses to wait, output will be displayed in the browser once fedex is finished (as shown in Figure 4). Otherwise, the user will be able to perform other browser tasks while fedex runs on the Server and, once the fedex process terminates, a job file containing its output will be created and stored under the user's ID. The form also requires the user to specify the file containing the EXPRESS specification. This file can be a user file or public file on the Server, or it can be an uploaded file.

The third page is created using form input from the second page. The content and the time required to display the third page is dependent on whether the user chooses to wait for the command results or not. If the user chooses to wait for the command results then the browser must wait for the

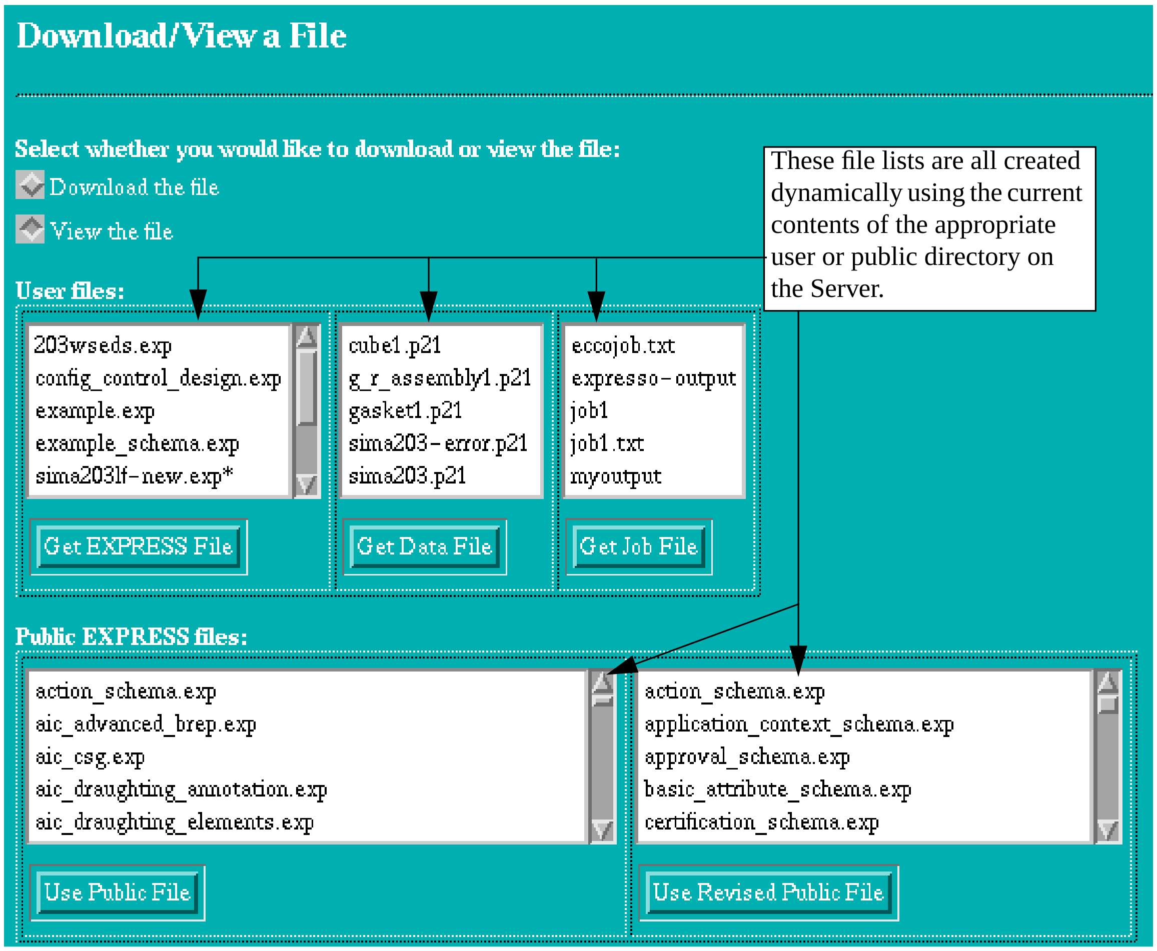
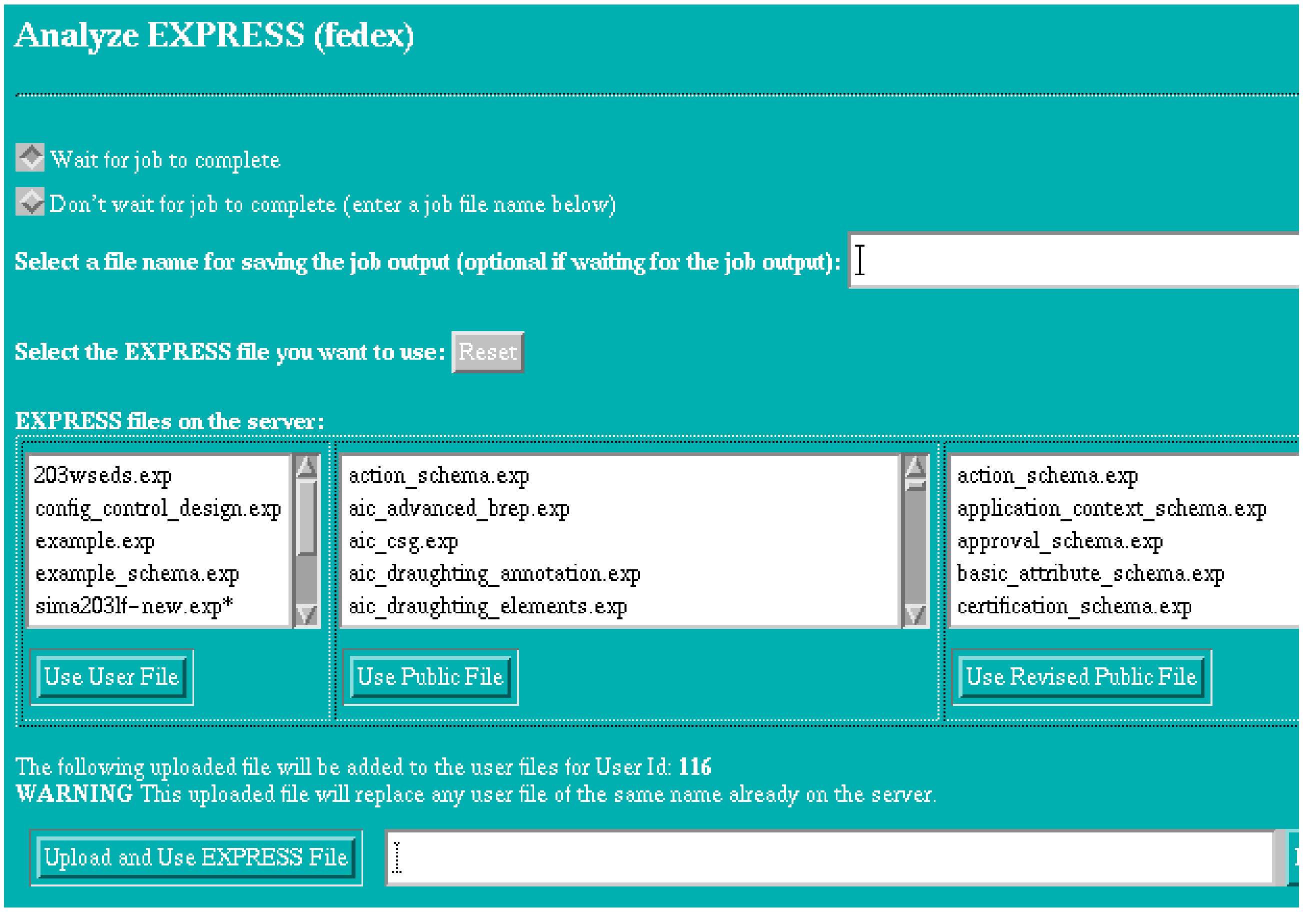


Figure 2. EXPRESS Web Server Input Page for Download/View

command to finish execution so the CGI script generating the HTML can finish telling the web browser what to display. When the command finishes execution, the CGI script generates the appropriate HTML containing the command results. As the HTML is generated by the script, the browser displays it for the user to see. If the user chooses not to wait for the command to complete he/she must indicate a file in which to save the command output. The CGI script tells the operatingsystem to start execution of the command and where to save the results.   It immediately



Analyze EXPRESS (fedex)

◇ Wait for job to complete

◇ Don't wait for job to complete (enter a job file name below)

Select a file name for saving the job output (optional if waiting for the job output):

Select the EXPRESS file you want to use: Reset

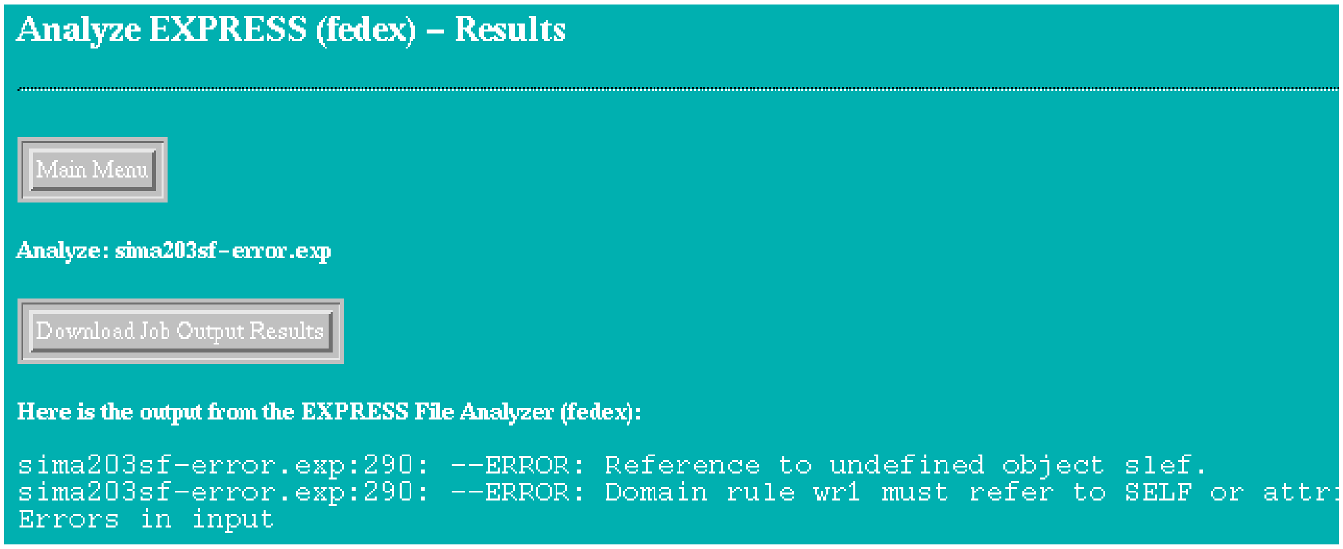
EXPRESS files on the server:

203wseds.exp	action_schema.exp	action_schema.exp
config_control_design.exp	aic_advanced_brep.exp	application_context_schema.exp
example.exp	aic_csg.exp	approval_schema.exp
example_schema.exp	aic_draughting_annotation.exp	basic_attribute_schema.exp
sima203lf-new.exp*	aic_draughting_elements.exp	certification_schema.exp
Use User File	Use Public File	Use Revised Public File

The following uploaded file will be added to the user files for User Id: **116**
WARNING This uploaded file will replace any user file of the same name already on the server.

Upload and Use EXPRESS File

Figure 3. EXPRESS Web Server Input Page for Analyzing EXPRESS Using the fedex Application



Analyze EXPRESS (fedex) – Results

Main Menu

Analyze: sima203sf-error.exp

Download Job Output Results

Here is the output from the EXPRESS File Analyzer (fedex):

```
sima203sf-error.exp:290: --ERROR: Reference to undefined object slef.
sima203sf-error.exp:290: --ERROR: Domain rule wr1 must refer to SELF or attr
Errors in input
```

Figure 4. EXPRESS Web Server Results Page Showing the Output from Analyzing EXPRESS Using the fedex Application

returns and generates a page indicating where to go to download or view the command results in the file specified by the user. Because of the dynamic nature of the above options, the third page (like the second page) must be generated using CGI and cannot exist as static HTML. Figure 4 shows a sample third page for when the user runs fedex and chooses to wait for the command output.

Several other applications besides fedex are available through the server. These include:

- The parser from ECCO [3], a freeware EXPRESS tool kit from the University of Karlsruhe in Germany. ECCO and fedex are useful together because each has its own strengths. For example, ECCO validates EXPRESS “WHERE” constraints while fedex does not, and fedex does a better job than ECCO handling external references.
- shtolo [11] - a software application developed at NIST that converts an EXPRESS specification containing pointers to definitions in the common building blocks into a normalized, self-contained specification.
- An application built using NIST’s Expresso tool kit [5] for validating data sets with respect to an EXPRESS specification.

The applications available on the Server are intended to cover some important needs of STEP standards developers. These are tools currently available to NIST that are feasible to install on NIST’s web server. They do not necessarily represent the best tools available for their intended purposes. Additional applications can be added to the Server as the need arises.

4. EXPRESS WEB SERVER DESIGN ISSUES

The Server's design addresses several challenges inherent in Internet applications. These challenges include:

- Giving users alternatives to deal with a slow Internet connection and time consuming Server commands.
- Providing a useful set of features without intimidating first-time or casual users.
- Striking a balance between HTML form simplicity and minimization of CGI script invocations.
- Combining the power of user IDs with the convenience of cookies.

To accommodate situations involving slow Internet connections or computing-intensive Server commands, users with a user ID are given the option of running commands in the background and saving the output to a file (see top of Figure 3). This enables them to avoid tying up the web browser while waiting for commands to finish executing. This feature would enable users to start multiple commands without waiting for each individual command to complete sequentially. The user is also given the opportunity to download the output displayed in the web browser after viewing it or to download the user-specified output file without viewing its contents. The download options free the user from having to rerun the application to view the output at a later time. To deal with an especially slow response time, the user could specify a file name for the application output and return later to download the file with minimal wait and annoyance. In the future an option may be added to allow output to be automatically emailed back to the user.

The user interface of the Server is designed to accommodate the needs of experienced users while not alienating novices. The Server may be used with or without a user ID. The latter offers the casual user the ability to go to the Server, enter and/or upload the necessary information, and immediately view and download the Server output. Without a user ID, temporary space is allocated on the Server as necessary and is immediately cleaned up. Without a user ID, certain options are not available such as upload of files to the server for reuse in subsequent operations. However, as users gain more experience with the Server, they may request a user ID so that they can take advantage of more advanced features offering flexibility, convenience, time-savings, and additional functionality.

The Server's web forms try to achieve a balance between ease of use, consistency, and efficiency. Forms are simple enough to be self-documenting and easy to use. In order to accomplish this, they contain extra buttons and, where necessary, Server commands are divided into simpler smaller steps. Extra buttons handle cases where differentiation between multiple selectable lists is needed (see the buttons in Figures 2 and 3). There are commands that require a choice of one file chosen from one of several directories. For purpose of documentation the files are not combined into one large list. Rather, each selectable list (labeled appropriately) indicates the files from the directory that may be used as input, and a corresponding button executes the command using the selected file from the appropriate directory.

Having separate forms for choosing a command (Figure 1) and supplying arguments to the command chosen (Figures 2 and 3) helps keep forms from being too complicated. Although having a single form for choosing a command and providing arguments would reduce the number of CGI scripts invoked, the resulting complexity of the form would outweigh any CGI overhead saved. Of course, once the Document Object Model (DOM) [22] and a standard web client scripting language [4] are fully supported by web browser vendors, it will be feasible to produce dynamic HTML documents, possibly eliminating the need for separate forms. The future impact of new web standards is discussed further in the next section.

As mentioned earlier, the Server issues user IDs in order to control access to private files. Similar functionality could have been accomplished through the use of cookies but with less flexibility. Consideration was given to providing every user with an unannounced user ID in the form of a cookie, an approach successfully used in another STEP-related web service [19]. Under this approach, disk space is allocated for the user during the first use of the service. Upon return to the service, unbeknownst to the user, the cookie is read to identify the user. A principal advantage to this method is that the user does not have to obtain, remember and enter a user ID. However, it has the following disadvantages:

- It assumes that a user's web browser accepts cookies and that users never (accidently) change or lose their cookie file.
- It assumes that users always use the same web browser (i.e. they never work from two different sites).
- User IDs cannot be shared among teams who wish to collaborate.
- Access control is dependent upon others not being able to read the cookie file; however, this is not necessarily the case if users share the same web browser configuration.

Because of these considerations, the Server only uses cookies as a convenience feature but does not require that users' web browsers accept them.

To summarize, the EXPRESS Web Server design uses ideas discussed above to maximize usability while aiming to minimize the possibility of failures due to differences between users' browser platforms. As a result, HTML pages are designed to be simple, with most of the complexity existing on the server rather than on the client. Although this approach may be criticized for failing to leverage some promising new web client technologies (discussed in the next section), it is consistent with recommended site design practices [16].

5. IMPACT OF EMERGING WEB TECHNOLOGIES

Emerging web technologies such as Java [8], XML (the eXtensible Markup Language) [1], and the DOM are creating new possibilities for interaction between users and Internet applications. The EXPRESS Web Server currently eschews these newer technologies in favor of older, more stable technologies such as CGI. However, as the new standards mature and web browser support for them continues to improve, these newer technologies may play a role in future enhancements to the Server.

One such desirable enhancement would be an interactive EXPRESS specification editor application integrated with the Server's EXPRESS tools and repository of public and private EXPRESS specifications. Such an application could:

- Save users the trouble of having to download, edit, and upload the relevant files each time they desire to make and test a change.
- Support incremental compilation and/or validation of EXPRESS code.
- Provide assistance with creating constraints needed to use an EXPRESS repository "building block" in an application-specific context.

An EXPRESS editor with the above functionality could be implemented using one or more Java applets. The applets would have to be able to provide the client-side processing necessary to allow users to interactively create EXPRESS definitions and write constraints. When necessary, they would make network connections to the Server host to obtain building blocks or run commands whose processing logic is too complex to encode in an applet.

Another desirable capability for the Server would be for users to be able to view EXPRESS code and STEP data sets in formats other than plain text. For example, it would be useful to be able to easily determine from looking at an EXPRESS specification normalized by shtolo, information such as which common building blocks are used in a particular definition or which building blocks are used most heavily. While an HTML document detailing such information could be generated using an EXPRESS parser and a post-processor [20], a preferable method in the future might be to translate EXPRESS specifications and data to XML, and have an XML-aware web browser render the specification using a style sheet. There is a good possibility that future versions of mainstream software tools such as word processors, spread sheets, and databases will be able to import data marked up as XML, making it feasible in some cases to integrate information represented using STEP with common business applications, thus lowering the cost of implementing STEP.

Another likely advantage of XML is that, once the XML linking and pointer languages [14] [15] are supported in web browsers, XML hyperlinking will be more powerful than what HTML allows. For example, it will become possible for EXPRESS specifications and data sets to reference specific portions of other documents on the web (using XPointers) and specify how the referenced data should be presented to the web surfer (using XLink's actuate and show attributes), enabling authors to tailor the referenced data's presentation to the referencing context. For example, data could be referenced in one context using a hot button to be clicked on, while in another context the same data could be seamlessly embedded into the web page currently being displayed.

Yet another web technology on the horizon that may influence the Server's future is WebDAV (World Wide Web Distributed Authoring and Versioning) [21], a set of extensions being developed for HTTP (the HyperText Transfer Protocol) to support remote web content authoring operations. The WebDAV extensions include capabilities such as overwrite protection and access control that, once implemented in mainstream web products, will provide functionality similar to the data access mechanisms in the Server. WebDAV also provides extensions such as version management that go beyond the Server's collaborative authoring capabilities. An added attraction of WebDav is that it is server-based rather than client-based, which should minimize browser incompatibility problems. If WebDAV achieves success as a standard, then future versions of the Server will be able to use WebDAV to provide more collaboration features than the current Server implementation allows. WebDAV may also be used by the server to provide better interoperability with other Internet applications.

6. CONCLUSIONS

The EXPRESS Web Server enables developers of STEP standards to build and populate EXPRESS specifications without having to obtain, install, and maintain a suite of EXPRESS software tools. It also facilitates collaboration among development teams through shared user IDs. The Server's user interface is flexible enough to accommodate the needs of both novices and power users. The EXPRESS Web Server's implementation currently favors established server-side web technologies such as CGI over newer, less-stable client-side technologies such as XML, scripting languages, and the DOM in order to ensure robust and consistent behavior across different web browser platforms. Once these emerging technologies mature, increased functionality and better integration with desktop software applications will be possible.

Although the Server was designed specifically for members of the STEP community, the issues encountered and addressed during its design are applicable to many other domains. The authors hope that other Internet application developers can gain from this experience.

7. ACKNOWLEDGMENTS

The authors would like to acknowledge Don Libes for his impact on this work. Don created cgi.tcl and the initial email-based EXPRESS Server. He also developed some of the EXPRESS applications installed on the Server. The authors also wish to thank NIST colleagues David Flater, Jim Fowler, and Karen Olsen, as well as the SAC Review Committee for their comments and suggestions.

This work is funded by NIST's Systems Integration for Manufacturing Applications (SIMA) Program.

8. REFERENCES

[1] Bray, Tim; Paoli, Jean; Sperberg-McQueen, C.M., Extensible Markup Language (XML) 1.0, W3C Recommendation, World Wide Web Consortium, February 10, 1998. <URL:http//www. w3c.org>

[2] The Common Gateway Interface, National Center for Supercomputing Applications, <URL: http://hoohoo.ncsa.uiuc.edu/cgi/>

[3] ECCO Tool Kit home page, Karlsruhe University, <URL: http://rpksun2.mach.uni-karlsruhe.de/~ecco/>

[4] ECMA-262, ECMAScript Language Specification, 2nd Edition, European Computer Manufacturers Association, August 1998. <URL:http//www. ecma.ch>

[5] Expresso Home Page, NIST, <URL: http://www.mel.nist.gov/msidstaff/denno/nist-expresso.html>

[6] ISO 10303-1:1994 Industrial automation systems and integration—Product data representation and exchange—Part 1: Overview and fundamental principles.

[7] ISO 10303-11:1994 Industrial automation systems and integration Product data representation and exchange—Part 11: Description methods: The EXPRESS language reference manual.

[8] Flanagan, David, Java in a Nutshell, 2nd Edition., O'Reilly, May 1997.

[9] Kristol, D.; Montulli, L., HTTP State Management Mechanism, RFC 2109, Internet Engineering Task Force, February 1997. <URL: http://www.ietf.org>

[10] Libes, Don, The NIST EXPRESS Toolkit: Introduction and Overview, NISTIR 5242 (NTIS PB94- 120664/AS), Gaithersburg, MD, October 25, 1993. <URL: http//www.mel.nist.gov/msidstaff/libes/>

[11] Libes, Don, Shtolo - Converting STEP Short Listings to Annotated Listings, NISTIR 5291, November 1993.

[12] Libes, Don, Concepts of the NIST EXPRESS Server, Proceedings of the First International Workshop on Services in Distributed and Networked Environments (SDNE), Prague, Czech Republic, June 27-28, 1994. <URL:http://www.mel.nist.gov/msidstaff/libes/>

[13] Libes, Don, Writing CGI Scripts in Tcl, Proceedings of the Fourth Annual Tcl/Tk Workshop'96, Monterey, CA, July 10-13, 1996. <URL:http://www.mel.nist.gov/msidstaff/libes/>

[14] Maler, Eve; DeRose, Steve, XML Linking Language (XLink), Draft, World Wide Web Consortium, 3-March-1998. <URL:http//www. w3c.org>

[15] Maler, Eve; DeRose, Steve, XML Pointer Language (XPointer), Draft, World Wide Web Consortium, 3-March-1998. <URL:http//www. w3c.org>

[16] Nielsen, Jakob, The Increasing Conservatism of Web Users, Alertbox (biweekly column), March 22, 1998. <URL: http://www.useit.com/alertbox/>

[17] Ousterhout, John, Tcl and the Tk Toolkit, Addison-Wesley, 1994.

[18] Raggett, Dave; Le Hors, Arnaud; Jacobs, Ian, HTML 4.0 Specification, W3C Recommendation, World Wide Web Consortium, April 24, 1998. <URL:http//www. w3c.org>

[19] STEP Tools Translation Service, STEP Tools, Inc., <URL:http://www.steptools.com/translate/>

[20] Varimetrix STEP Page, Varimetrix Corp., <URL: http://www.vx.com/Other/STEP/step.html>

[21] Whitehead, E.J., Jr. and Wiggins, M., WEBDAV: IETF Standard for Collaborative Authoring on the Web, IEEE Internet Computing, September-October 1998, pp. 34-40.

[22] Wood, Lauren, et al., Document Object Model (DOM) Level 1 Specification, Version 1.0, W3C Proposed Recommendation, World Wide Web Consortium, August 18, 1998. <URL:http//www. w3c.org>