NTIA Report 99-365

A Search Engine for Federal Standard 1037C

William Ingram
Evie Gray

U.S. DEPARTMENT OF COMMERCE
William M. Daley, Secretary

Larry Irving, Assistant Secretary
for Communications and Information

March 1999

CONTENTS

page

List of Figures

page

A Search Engine for Federal Standard 1037C

William Ingram and Evelyn Gray[1]

Abstract. Federal Standard 1037C, *Glossary of Telecommunication Terms,* contains more than 5800 entries defining the components of several disciplines in telecommunications technology. Use of this large glossary is enhanced with the search engine designed specifically for the two HTML versions of the glossary: the online version and the CD- ROM version. The search engine provides speedy access to the entire text of the definitions containing the specified character string used in the search. This report discusses the need for the search engine, its design criteria, functions, enhancements, and benefits. The computer code for the search engine, which was developed using Perl scripts, is listed in the appendix.

Key words: CD ROM; Federal telecommunications standards; glossary; HTML; hypertext; Internet; NII; search engine; Web

1. BACKGROUND

From 1974 to 1996, the Institute for Telecommunication Sciences, under the sponsorship of the NCS, has had a leadership role in developing and updating Federal Standard 1037 (FED-STD-1037C), *Glossary of Telecommunication Terms,* and its predecessor documents. This paper discusses the development of the search engine for the hypertext versions of that Standard.

1.1 Purpose of Federal Standard 1037C

Federal Standard 1037C, *Glossary of Telecommunication Terms,* advances interoperability of telecommunications equipment and services by standardizing telecommunications terminology. The standard supports the fastest growing segment of our economy—the technology segment—and it benefits U.S. telecommunications companies on the world stage while promoting international interworking of importance to U.S. trade.

Standards like FED-STD-1037C are most useful if universally accepted; wide distribution and ease of use strengthen and enhance acceptance of a standard. The standard was developed in hypertext form and placed on the World Wide Web (Web or WWW) to aid in its ease of use and wide distribution.

A search engine was developed for FED-STD-1037C to enhance the ease of use of the large glossary. This report documents the development of the search engine and its usefulness.

[1] The authors are with the Institute for Telecommunication Sciences, National Telecommunications and Information Administration, U.S. Department of Commerce, Boulder, CO 80303-3328.

1.2 About FED-STD-1037C

Federal Standard 1037C, *Glossary of Telecommunication Terms*, containing more than 5800 entries, was published in hard copy (400 pages) and on the Web in 1996 to serve the Federal Government, industry, and academia by providing standardized language necessary for acquisitions, contracting, research and development, operations and maintenance, and the clear communication of technological developments and advances. The work of developing the standard, its Web pages, and—later—its search engine was done by the Institute for Telecommunication Sciences (ITS), the research and engineering arm of NTIA, under the sponsorship of the National Communications System (NCS).

The standard is widely used by telecommunications industry professionals, both in the United States and overseas, and it is frequently cited in acquisitions documents. The FED-STD-1037C Web site has received more than 148,000 hits in the first 27 months since its creation and continues to receive approximately 3,000 hits per month.

1.2.1 Scope of the Telecommunications Glossary:

The glossary includes standardized definitions in the fields of

- ✔ Communications Security
- ✔ Data Processing
- ✔ Facsimile
- ✔ Fiber Optics Communications
- ✔ Grounding and Bonding
- ✔ National Security/Emergency Preparedness
- ✔ National Information Infrastructure (NII—the "Information Superhighway")
- ✔ Networks (including Internet, LANs, MANs, WANs, Intelligent Nets, ISDN, Broadband ISDN, Network Architecture, and Network Management)
- ✔ Premises Wiring
- ✔ Radar
- ✔ Radio Communications
- ✔ Spectrum Sharing
- ✔ Telegraphy
- ✔ Telephony
- ✔ TV (UHF, VHF, Cable, and HDTV)

1.2.2 Anticipated Users of the Glossary

The glossary, whose use is mandatory for Federal agencies, is designed for the following users:

- ✔ Purchasing agents
- ✔ Government standards writers and users
- ✔ NII planners
- ✔ Testing and operations and maintenance workers
- ✔ Research and Development workers
- ✔ Technical writers
- ✔ Telecommunications designers, vendors
- ✔ Telecommunications instructors

1.2.3 Glossary Sources (for technical content and for additional copies)

Definitions were collected from several Government agencies and from work of the ANSI[2]-accredited X3.K5 vocabulary subcommittee [1] (now called the "NCITS.K5"—National Committee for Information Technology Standards—vocabulary subcommittee), whose members also participate in the work of the international ISO (International Organization for Standardization) vocabulary

[2] ANSI is the American National Standards Institute.

committees [2]. Printed copies and CD ROMs of the glossary may be obtained free of charge from:

National Communications System
Attn: Ms. J. Orndorff
701 South Court House Road
Arlington, VA 22204-2198

The hypertext version (in HTML), the PDF (portable document format) version, and a version in word-processing format are available on the CD ROM and are also directly available at the Web address:

http://glossary.its.bldrdoc.gov/fs-1037

Web access is also available through hypertext links to several other Web pages such as the Web page for the sponsoring agency, the National Communications System:

http://www.ncs.gov

and the Web page for the Institute for Telecommunication Sciences:

http://its.bldrdoc.gov

1.2.4 History of the Glossary's Development

For more than two decades, the Institute for Telecommunication Sciences (ITS) has had a leading role in the development of the 1037-series of Federal Standards. For the 1037C edition of the standard, ITS staff chaired the Subcommittee to revise the standard, and ITS served as the senior technical editor, the developer of the HTML version, and the designer of the search engine for the standard.

1.2.4.1 Hypertext Development of the Glossary

Staff at ITS developed the hypertext version of FED-STD-1037C by using Perl scripts to insert automatically many thousands of required links to related definitions and the respective color illustrations. Details of this development are discussed in a report [3] by Ingram and Gray and in a conference paper by Gray and Ingram [4].

The hypertext version of the standard accommodates most computer platforms and a variety of hypertext browsers. The display of the hypertext document is best if viewed with a browser that supports "frames," which is a viewing screen or window that is divided into smaller windows, one of which is much larger than the others. A licensed copy of such a browser is supplied on the CD ROM containing the HTML version of FED-STD-1037C.

1.2.4.2 Advantages of Hypertext for the Glossary

In many ways, the *Glossary of Telecommunication Terms* is ideally suited for use in the hypertext format. These instances of suitability coalesce to provide significant enhancements to the ease of use of the glossary.

1.2.4.3 Index-card format

In hypertext, one of the most useful and inviting display formats is a computer screen with a single idea or a single paragraph on it. This format has been described as an "index-card format," because of the limitations in size and because of the fact that a single idea is presented on a screen of its own.

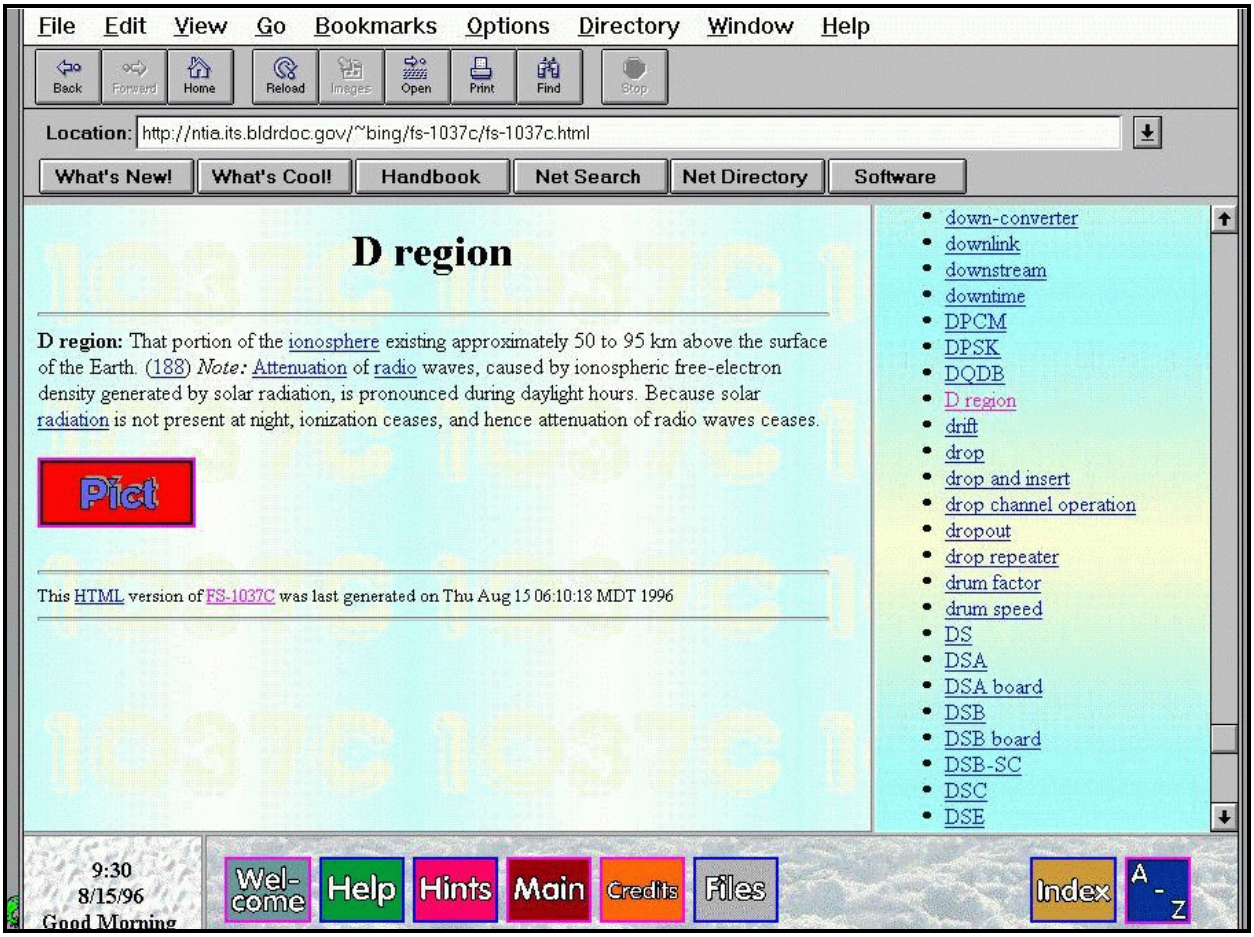

Figure 1. Full display of the term *D region*.

The hypertext glossary displays a single definition in the central (largest) frame on the screen. The focus provided by this display technique is helpful and sharp. Figure 1 illustrates a typical computer screen of the HTML version of the glossary, in this case, the computer screen that appears when the user clicks on the definition for *D region*, (a layer of the ionosphere that is important to telecommunications).

The *D region* definition was accessed by clicking the *D region* term in the hypertext list of terms (underlined) in the right-hand column of the screen.

The illustration accompanying the definition of *D region* is accessed by clicking on the red "Pict" (picture) icon below the definition. The reason that this thumbnail Pict icon is provided rather than the full picture is that the transmission of the full picture could take a long time on a slow connection (*i.e.*, a modem). By using a Pict icon, viewers uninterested in the picture are able to access

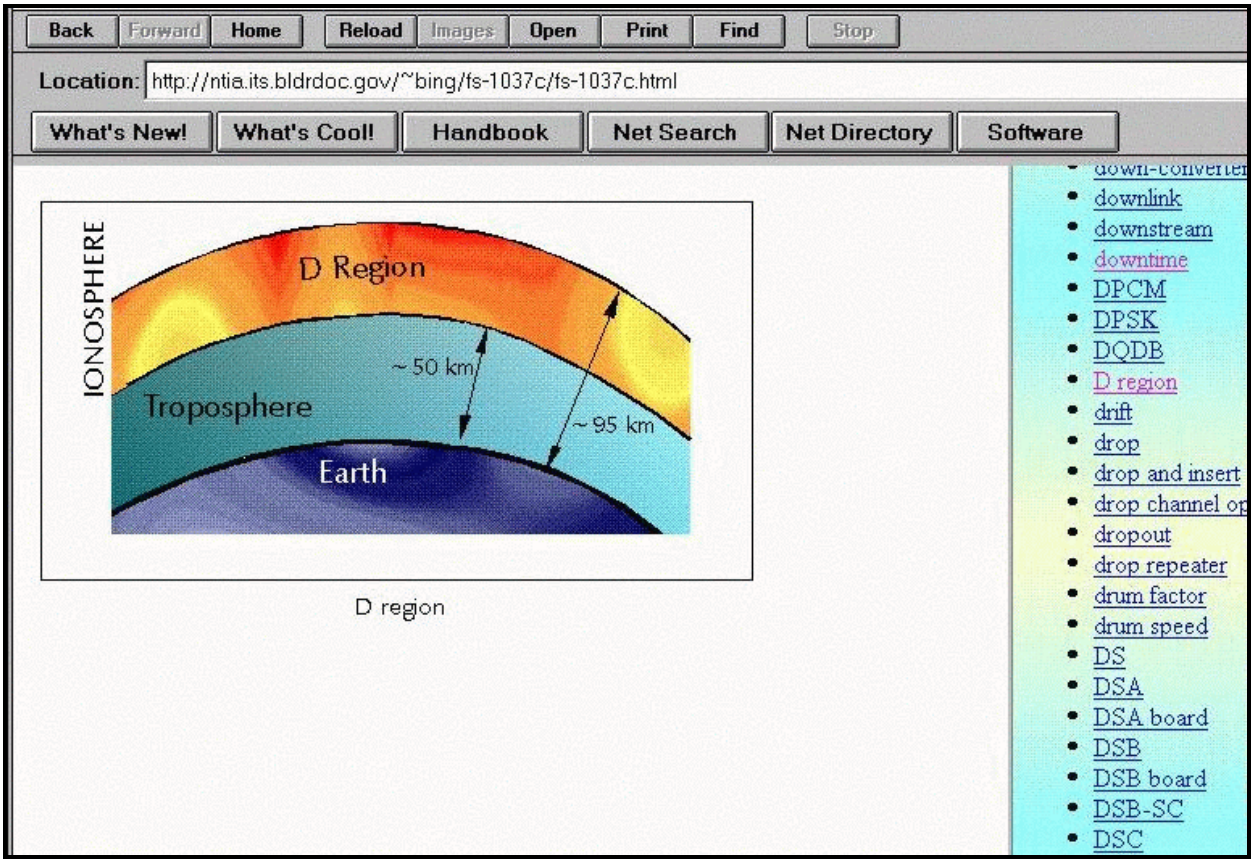

Figure 2. Full display of an illustration after clicking the "Pict" icon.

the text of the *D region* definition quickly. Anyone interested in the illustration can view it in full frame by clicking on the Pict icon. When they do, they will see the picture that is shown in Figure 2. It will appear in the central frame of their computer screen, replacing the definition of *D region*, but leaving all of the information in the other frames accessible.

1.2.4.4 Nonlinear reading

Almost no one reads a dictionary from cover to cover, from A to Z. Most readers browse for the definition of a word or words of interest at that moment. Often the readers will look up related words; rarely will they spend time on adjacent words unless the adjacent words are related to the initial word of interest. The hypertext Glossary allows the reader to focus on the initial definition of interest and to browse related definitions by clicking on hyperlinks indicated (in blue, with an underline) in the initial definition.

1.2.4.5 Speed of access

The hypertext version of the Glossary provides rapid, efficient access to the contents of this large glossary. In hard copy, the user is required to turn a precise (and often large) number of pages to access a specific definition. The hypertext version allows much faster access with the click of a mouse button. A

search engine is seen as a mechanism for enhancing this access.

1.3 Search Engine Development

The most recent effort has been to develop a search engine to be used with the hypertext versions of the Glossary (*i.e.*, with the Web HTML version and with the CD-ROM HTML version).

Commercially available search engines did not provide sufficient flexibility for use with FED-STD-1037C, so the editors developed a custom search engine that searches the complete online text of all 5800 definitions and then displays an ordered list of results in less than 9 seconds. The search engine produces identical results when searching the CD ROM version of the glossary, but the results may be slower and the speed of the process depends on the computer equipment used for the search.

The CD ROM of FED-STD-1037C contains the necessary support software to run the search engine to perform searches on the CD. The CD-ROM search engine for the HTML version of the standard is designed specifically to work in Windows 95®, 98®, and NT environments.[3] No other platforms are supported by the search engine that is contained on the CD ROM.

[3] Certain commercial equipment, instruments, or materials are identified in this paper to specify adequately the development of the glossary and its use. In no case does such identification imply recommendation or endorsement by the National Telecommunications and Information Administration, nor does it imply that the material or equipment identified is necessarily the best available for the purpose.

1.3.1 Running the Search Engine

The search engine is easily accessed in the online version of the Standard by performing the following steps.

- ✔ Click on the words “Search Engine” in upper right portion of the glossary's initial screen.
- ✔ Type the word(s) for which you would like to search in the “Phrase to search for” box. [This text is *not* case sensitive.]
- ✔ Indicate whether you wish all hits to be listed, or only the top 25 hits or 100 hits. [The default value is the top 25.]
- ✔ Press the “Enter” key or click on the “Find It” button.

To exit the search engine and return to the main glossary, click on the underlined word “main,” which is found in the lower frame; then, click on the “non-search engine version of FS-1037C” words that appear in the new central frame.

The FED-STD-1037C search engine observes the standard search-engine conventions regarding use of quotation marks (“ ”) to search for character strings jointly or separately.

Figure 3 illustrates the results of executing the above four steps to perform a search for the word “telephone” in the online HTML version of the glossary.

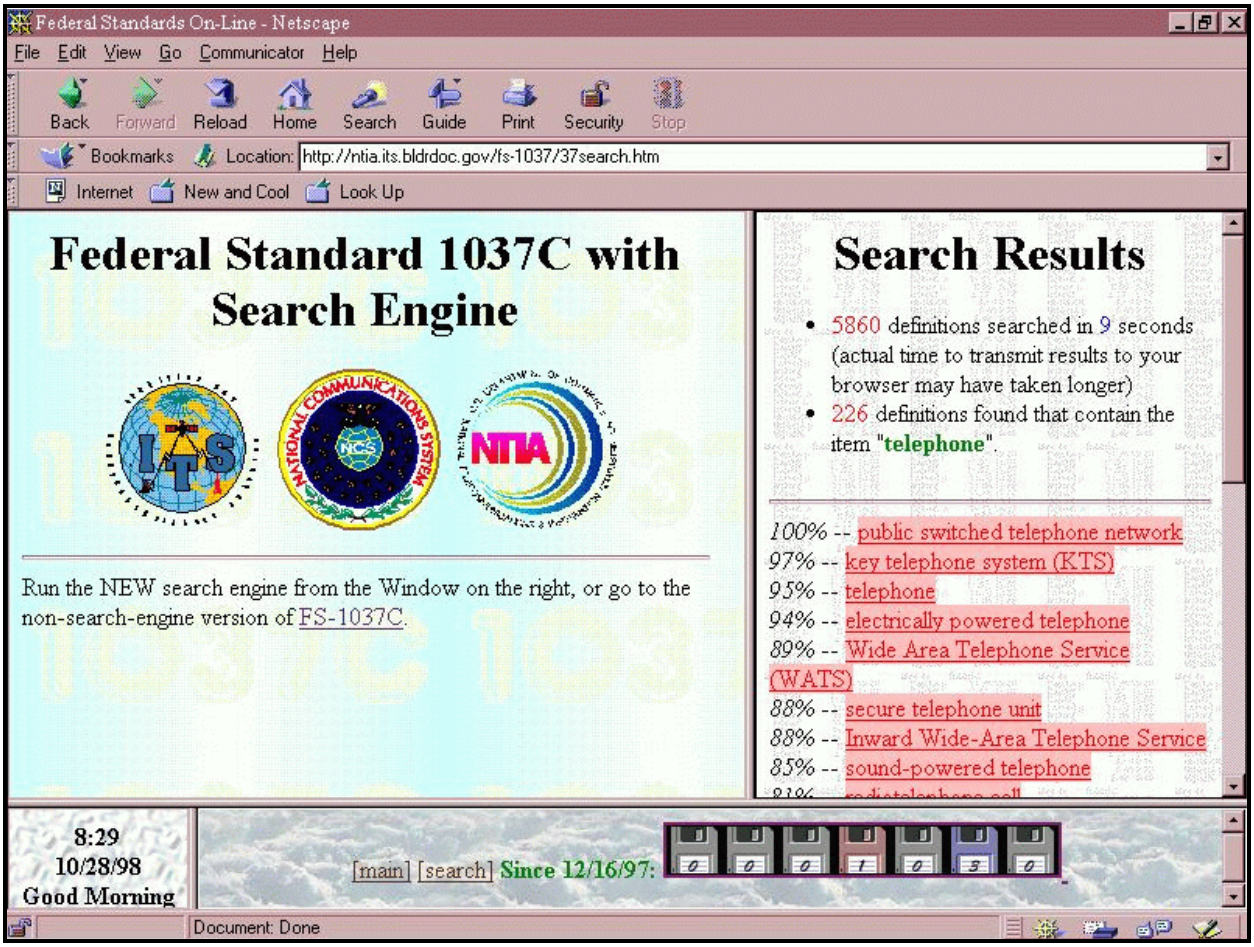

Figure 3. The screen displaying the results of a search for the word *telephone* in the HTML online version of FS-1037C.

The resulting list of definitions use color codes to indicate the likely relevance or importance of each hit to the search word ("telephone" in the case of the example). In most cases, a pink background around a listed "hit" indicates that the pink hit contains the searched-for word in its title (its term name) as well as in the text of its definition. A green background means that the search word appeared multiple times but in the definition only, not in the term name. A yellow background color indicates that the search word appeared in the definition, but only once. The assignment of pink, green, and yellow color backgrounds is roughly as stated above, but the actual color assigned to any given hit may differ from this scheme because of the weightings assigned. See Table 1 for exact weighting factors used in the algorithm to calculate the background colors.

In Figure 3, the top several of the 58 definitions where the word "telephone" was found are listed in the right-hand frame. All of these listed entries in this frame are hyperlinked so that a single click of the mouse

will take the reader immediately to that definition.

1.3.2 Limitations of the Search Engine

Users of the CD-ROM version search engine must be aware that the search engine will support PC Windows 95®, 98®, and NT platforms only, while the online search engine is compatible with nearly all of the popular platforms available today. The CD-ROM version of the search engine is also designed so that it is not case sensitive.

Users of the CD-ROM version of the search engine must install the search engine precisely as instructed in the "README.TXT" file supplied on the CD ROM. Configuration of the computer and assignment of the CD ROM drive are other critical parameters for the successful execution of a CD ROM search of FED-STD-1037C.

Further, users of the CD-ROM version of the search engine must be aware that searches on the CD may be slower (depending on the equipment used) than searches of the online version of FED-STD-1037C, even when using the same character-string search.

The user can tailor the search and enhance its precision by typing more than one word in the "Phrase to Search for" box. For instance, if the user types "hf telephony" in the search box, the results will list only those entries containing both the words "HF" and "telephony." However, results are appreciably slowed (in either the online or the CD searches) if more than one word is used in the search.

1.4 Benefits of the Search Engine

While the hypertext links and the hypertext index allow the user to reach a desired definition more quickly than is possible with hard copy, the search engine for FED-STD-1037C allows the user to reach *all* pertinent definitions quickly. This capability provides:

- ✔ a more tailored, comprehensive use of the glossary,
- ✔ a more accurate survey of relevant contents, and
- ✔ a more thorough review of the current edition, in preparation for an updated edition.

The hypertext index is of some assistance in each of these three endeavors, but the search engine has the further advantages of:

- ✔ ranking—by likely relevance—of the results of the search,
- ✔ speedier results in collecting a list of *all* related words, and
- ✔ absolute thoroughness in executing the search. (The hypertext index is only an index of the principal families of defined terms; the search engine searches the entire text of all terms.)

Precision is a further benefit provided by the capability of executing a multiword search. Being able to hone the search (using more than one word in any individual search) provides exactness and meticulousness that can more than compensate the user for the extra time required to complete the multiword search.

The search engine speeds and facilitates review of the entire glossary, which is an advantage for any reviewers who may eventually develop proposals for revising the glossary.

1.4.1 Development of the Search Engine

Because commercially available search engines were not optimal, ITS developed two customized search engines, one that was designed specifically for the Web hypertext version of FED-STD-1037, and one for the CD-ROM hypertext version of the same glossary. Details of development are outlined in Section 2 below. The search engines were developed using Perl scripts and related support server software. The two search engines provide identical, ordered results in an easy-to-view frames format that allows the user to see the ranking of each "hit." The user is able to click on any of the hyperlinked term names listed as "hits" (or results) at the completion of the search engine's work.

The search engine for the Web version of the glossary was completed and tested first, followed by the search engine for the CD ROM. Both were designed to have the same look and feel and to produce exactly the same results (in the same format) when used on identical files of the glossary. The development of the CD-ROM search engine was more exacting and more complex than the search engine for the Web because configuration files can vary widely among individual computer systems.

1.4.2 Users of the Search Engine

In the first 6 months since posting it on the Web, the search engine was accessed several thousand times. A study of the list of accesses to the site shows that, like Web accesses of the glossary itself, users of the search engine for the glossary include government users, academia users and industry users. Users are able to tailor their searches to suit specific needs of the moment and to refine or modify the searches.

1.5 Web Version of FED-STD-1037C

1.5.1 Development

The ITS staff developed a semi-automated system to create the 5800 Web pages for FED-STD-1037C online. The system relied on several levels of input files. The first level created a Web page (otherwise known as a "computer screen") for each term and definition.

The next level searched the text of each definition and built links to any defined word that appeared within that text. For example, the definition of the term *radio control* contained the defined terms *radio* and *station*. Hypertext links were automatically created to allow the Web user to click on either of these words in the definition and to have the respective definition appear on the screen.

A final level of hyperlink development was created manually. This level listed numerous exceptions for the automatic compiler to consider. For example, the term *branch* appeared more than 30 times within the 5800 definitions in FED-STD-1037C (once as a term name). But in 25 of those appearances, the context of the definition made it clear that the term was not being used in the same sense that *branch* was defined in FED-STD-1037C. In most of those cases, it was used as in "a *branch* of science." It would have been misleading to the Web user to build a hyperlink from such a use of the word *branch* to the defined term *branch*, which defines a leg of a network.

Considerable time and effort was expended to convert the graphics, equations, and special characters (*e.g.*, Ω, ∞, and Å) into formats that could be displayed as part of a Web page using the most up-to-date HTML techniques available at the time.

1.5.2 Users and Benefits

In the first six months that FED-STD-1037C was available on the Web, the page was accessed more than 20,000 times. Over the first 27 months, that number has grown to more than 148,000 accesses.

The glossary is used primarily by Federal departments and agencies. It was anticipated that the glossary would be used by Government procurement agents and Federal engineers developing Federal and international standards. An analysis of the first 90,000 hits shows that industry and academia are also heavy users of the glossary.

1.6 The Web As The Future For Federal Documents

The World Wide Web allows users to instantly access huge volumes of data worldwide. Users need only a computer and Internet access to view any of the millions of documents or Web pages.

1.6.1 Federal Standards

Federal standards are a valuable resource for Federal, state, and local users; academia; and industry. The usefulness of Federal standards can be lessened by the inconvenience of traditional hard-bound publishing, making it difficult to quickly locate all of the subsections of a standard that are relevant to a particular subject of interest.

By placing Federal standards online as Web pages, search engines can be used to quickly isolate and present occurrences of key words and phrases.

1.6.2 Other Documents

One can argue that there are many advantages to making most Government documents instantly available over the Internet. The advantages of Web distribution include:

- ✔ HTML links to related documents.
- ✔ Reduced printing costs.
- ✔ Improved clarity (with the addition of color).
- ✔ Searchability of the entire document (with the addition of a search engine).
- ✔ Collection of e-mail comments (a possible added feature).
- ✔ Wider review of the document.

2. DESIGNING THE SEARCH ENGINE

After the Web version of FED-STD-1037C was online for several months, user feedback indicated that it would be helpful to be able to search the entire glossary for specific words and phrases. At that time, the glossary hyperlinks provided the capability for locating specific terms only by name or by subject. If the user knew the term name, the user would select the appropriate letter of the alphabet and then scroll through the term names beginning with that letter until the desired term was located.

The user could select Appendix B of the glossary and see a limited list of families of term names. For example, Appendix B displays hyperlinked lists of all of the definitions in the

family of *radio* terms. Under that family listing are 122 term names related to *radio* (including the term *radio* itself). These 122 terms were selected by the editors of FED-STD-1037C to be the most appropriate terms to be included in that family. The user can click on and select for viewing any of those term names from that index list in Appendix B of the glossary.

Some users may, for example, want to search the text of every definition for the word *radio*. Using the search engine, this kind of search would produce 397 definitions that contain the word *radio*. To accommodate that more exhaustive type of review, ITS developed a search engine for FED-STD-1037C and included access to it on the FED-STD-1037C Web page.

2.1 Development

2.1.1 The Layout

The concept of a search engine specifically designed to search a series of Web pages has been around since 1992, when several such services became accessible via the Web. Most casual users of these Web search engines have come to expect a certain format. The ITS staff designed the search engine for FED-STD-1037C to offer features and functionality that are familiar to users of established Web search engines.

Existing search engines could not be efficiently used for searching FED-STD-1037C because most search engines will simply search for occurrences of the target phrase and will create links to any Web page with the matching phrase. While this approach uses a simple, adequate search algorithm, the editors of FED-STD-1037C wanted to perform more meaningful searches. Some search engines will count the number of occurrences of the target phrase and order the results accordingly, providing a better list.

> **computer: 1.** A device that accepts data, processes the data in accordance with a stored program, generates results, and usually consists of input, output, storage, arithmetic, logic, and control units. **2.** A functional unit that can perform substantial computation, including numerous arithmetic operations or logic operations, without human intervention during a run.

Figure 4. The definition of the term *computer*.

However, if a user performs one of these kinds of searches of FED-STD-1037C for the word *computer*, one of the most meaningful matches should be the actual definition of the term *computer*. Unfortunately for these basic search engines, when FED-STD-1037C was written, the authors adopted a policy of not using a term name within its own definition (the rationale being that no useful information is provided by using a term name to define itself). This restriction means that the word *computer* appears in the FED-STD-1037C definition of the term *computer* only once (see Figure 4), as the title (or "term name") of the definition. But the word *computer* appears 285 times in other definitions.

2.1.2 Weighting

A generic search engine would not assign any special significance to the Web page found by the example in the previous section, even though it is, in fact, the only page containing the definition of the term. The ITS wanted to design a search engine that would be able to assign extra significance (*i.e.*, a weighting

factor) to matching occurrences of the target phrase where appropriate. For example, when searching for the word *radio*, it is a better match to find the word *radio* in the definition than to find the word *radionavigation*. Additionally, finding the word *radio* in the title of the definition would indicate an even more significant match. Table 1 shows some of the weighting factors used in the ITS search engine.

Table 1. Weighting Factors for the Search Engine for FED-STD-1037C

Relative Weight	Occurrence Within a Definition
2	Phrase was found.
5	Phrase was found as a separate whole word (*i.e.*, *radio* as opposed to *radionavigation*).
10	Phrase was found within the title of the definition.
30	Phrase is the entire title of the definition.
10	All parts of a multi-part search request were found within the definition.

Further, the relative weight of an occurrence of the target phrase is modified by how far into the text of the definition the match occurs. For example, if the match occurs at the beginning of the text, the match is assigned a greater weight than if the match occurs at the end of the text. In Figure 5, the word *noise* appears nine times. The first occurrence of the word (after the title) appears approximately 5% of the way through the text of the definition. Therefore, that match is assigned a value of 95% of the maximum possible value. The second occurrence of the word appears approximately 15% of the way through the text and would receive only 85% of the maximum value.

These percentages factor into the weightings that are displayed with the FED-STD-1037C search results, as seen in Figure 3.

channel noise level: 1. The ratio of the channel noise at any point in a transmission system to an arbitrary level chosen as a reference. *Note 1:* The channel noise level may be expressed in (a) dB above reference noise (dBrn), (b) dB above reference noise with C-message weighting (dBrnC), or (c) adjusted dB (dBa). *Note 2:* Each unit used to measure channel noise level reflects a circuit noise reading of a specialized instrument designed to account for different interference effects that occur under specified conditions. **2.** The noise power density spectrum in the frequency range of interest. **3.** The average noise power in the frequency range of interest.

Figure 5. The definition of the term, *channel noise level*.

Figure 5 also shows one of the potential pitfalls of designing a search engine that assigns weights to occurrences. The word *noise* appears in this definition nine times. Figure 6, the actual definition of the word *noise*, shows only one match. A traditional search engine will perceive the nine matches in the first definition as an indication that that definition (with nine matches) is a more appropriate match for *noise* than the second definition (with a single match). In many cases,

this disparity exists even if a large weighting factor is assigned to matches within the title. For example, if the values in Table 1 are used to weight a search for the word *noise*, within Figure 5 and Figure 6, the totals would be:

- ✔ *channel noise level*: 1 match in title, 10 points. 8 matches in body of definition, 5 points each. Total: 50 points.

- ✔ *noise*: 1 match as the entire title, 30 points. Total: 30 points.

Such a search would show that *channel noise level* is a better match. Obviously, if a user searches for *noise*, the actual definition of *noise* should be the best match. To help offset this imbalance, the ITS search engine considers only the first four occurrences of a target phrase when analyzing a match.

noise: 1. An undesired disturbance within the frequency band of interest; the summation of unwanted or disturbing energy introduced into a communications system from man-made and natural sources. **2.** A disturbance that affects a signal and that may distort the information carried by the signal. **3.** Random variations of one or more characteristics of any entity such as voltage, current, or data. **4.** A random signal of known statistical properties of amplitude, distribution, and spectral density. **5.** Loosely, any disturbance tending to interfere with the normal operation of a device or system.

Figure 6. The definition of the term *noise*.

2.1.3 Display

By designing its own search engine, ITS can control the display of the results. Many commercial search engines include advertisements in their results to help offset computer and Internet costs. Such advertisements are not appropriate for U.S. Commerce Department Web pages.

It is common for the operators of many commercial search engines to skew the search results towards certain Web pages, either because those Web pages are considered to be better repositories of information than others or because of other unrelated commercial considerations.

2.1.4 The Format and Style

The staff at ITS examined and used many commercial and free search engines over the Internet in the months before the ITS search engine was developed. The ease of use of each was noted, along with any interesting and useful features. Some of these features were incorporated in the ITS search engine. For example, many search engines display a confidence factor next to each matching result. This confidence factor indicates how well the displayed Web page matches the target search phrase. The ITS staff incorporated a confidence factor into the FED-STD-1037C search engine (by using weighting).

Most search engines also will limit the number of matching entries displayed to the user. If a user accidentally searched for a common word or phrase (the word *the*, for example), thousands of results would be displayed. The user would be overwhelmed by such a list and would extract no useful information from it.

Many search engines will limit the display to the first ten matches, or will allow the user to choose a maximum number of matches to display. The search engine for FED-STD-1037C gives the user three choices for the

number of results to display: the first 25 matches, the first 100 matches, or all of them. If the user does not make a selection, the first 25 are displayed.

2.2 Programming

Programs for the HTML environment can be either server-side or client-side. Server-side programs are run on a Web server computer and results are sent to the Web user. Client-side programs are included with the Web page when it is first downloaded and run when needed. The advantages and disadvantages of each type of program are discussed in the section below. Also discussed in this section are aspects of using Perl for developing the computer code.

2.2.1 Server Side

Server-side programs are located at a Web server, and have direct access to data files that a Web user is not able to view. Using this method, sensitive data can be processed by the server-side program and only the relevant data will be sent to the Web user. For example, if a user of a commercial Web site wants to view the books that he or she had purchased over the last year, the user would ask for that information and would provide his or her account number and password. A server-side program would then process the request and, if the password matches the account number, would return a list of books recently purchased by that user. The user, however, would not be able view a list of books purchased by any other user, since the master data file, containing all of the purchase records, is not accessible by Web users.

Authors often use server-side programs to restrict access to files that do not necessarily contain sensitive data. Sometimes the data are coded or too confusing for a Web user to understand. Other times, a data file may be too massive to send to a user. The server-side program itself is never sent to the user, so that proprietary techniques and codes can be used without concern that the contents of the program will be viewed or used without permission of the author.

Since the Web server loads and runs the server-side program each time a remote user accesses it, the server is always loading the latest copy of the program. Changes that a program author makes will become immediately available to Web users. Similarly, since the server-side programs access the most current data files when run, the latest data are returned to the user.

This immediate access to programs and files can be a disadvantage as well. If the author of the program introduces an error into the program while in the process of updating it, users will be unable to access the correct data until the error is corrected. Similarly, if the administrator of the data files incorporates incorrect or incomplete data into the data files, the Web user will receive incorrect information until the problem is corrected.

This consequence leads to another disadvantage. A user cannot have the confidence that, for example, running the same server-side program two days in a row will yield the same results. The data files or the programs themselves may have been changed between accesses. Or, the programs or data files may have been completely removed from the server so that they are no longer available. This variability is generally not a concern with

FED-STD-1037C because, by convention, the standard is only updated approximately every 5 years.

The greatest advantage to using server-side programs is that the program author does not need to worry about the client's setup. Every user who accesses the Internet with his/her own browser is likely to have a distinct computer configuration. It would be an immense task to design one piece of software to run correctly on any random configuration. However, since server-side software is not intended to run on the client's machine, the software only needs to be designed to run correctly on the one server.

Server-side programs are often called Common Gateway Interface (CGI) scripts. The primary computer languages used to write such scripts are C, C++, and Perl, although almost any language can be used.

2.2.2 Client Side

Client-side programs are those that are designed to run on a client's own machine rather than on a Web server. The Web user must download the desired program before it can be used. In most cases, the program is embedded within a Web page and is automatically downloaded when a user accesses that Web page. Such downloading can require a long time. For sufficiently complicated programs, the embedded code can be ten or a hundred times longer than the code for the Web page itself.

However, once the embedded software is downloaded to a client's machine, it can be run multiple times without further downloading. The user can even disconnect from the Internet while running the program. Additionally, the user can save the Web page (which contains the embedded program) to a hard drive and can use it again later without downloading it. However, if a user does save and reuse an embedded program, the user is not assured of using the latest version available. The ability to save the program to disk may allow the user to copy the program, to modify it, and to use it for other purposes. This capability can be an advantage (because it offers flexibility), or a disadvantage (because it may introduce uncontrolled variations), depending on the extent and nature of the modifications.

Client-side applications are most often written with the Java programming language, which is an object-oriented computer language and environment based on C++ that allows for the incorporation of moving text, animation, and interactive games into a Web site. Small procedures called applets, written in Java may be downloaded through the Internet and run directly by an application (often a Web browser) on the client's computer.

The Java language is specifically designed so that a program can work properly under several different operating systems, without the need to convert the program.

The fact that client-side software runs on the client's machine, rather than the server machine, has several disadvantages in this application. The first is that the Java program will not have access to any file that is not installed on the client's machine. The second disadvantage is that the Java code has access to any file on the client's machine. This can be dangerous if an unscrupulous Web-page designer includes a malicious program in his or

her Web page. When an unsuspecting user accesses such a page, Java code can begin running, without the user's knowledge or consent. This Java code can access any file on the user's drive, changing or deleting important files or sending copies of private files on the hard drive to remote machines.

Fortunately, many browsers have options to prevent Java programs from running or from accessing files on the user's hard drive. Others notify the user before Java code is run. Unfortunately, many computer users are not aware of these options.

On the other hand, those users who have prevented all client-side programs from running will prevent any useful and benign Java programs from running as well. The user must consider the advantages and disadvantages of disabling client-side applications.

2.2.3 Perl

After weighing the advantages and disadvantages listed previously, ITS decided to create a server-side application. Practical Extraction and Report Language (Perl) was selected for writing the server-side programs.

Perl was developed as a computer language to easily manipulate text, files, and processes on a UNIX™ computer. Perl was later converted for use on a wide variety of computer operating systems. Perl is, in many ways, a simple language. The types and structures used by Perl are easy to use [5] and understand. But, Perl is also a rich programming language. It provides many subtle processing techniques to produce quick results, and it is considered to be a language that is appropriate for small-to-medium programming tasks.

Perl is compiled at run-time, unlike programs written with C and similar languages, in which the source code is compiled once after it is written or updated.

When source code is compiled immediately after editing, the resultant program is an executable program. The executable program can be copied between computers with similar operating systems. In this way, program authors can distribute executable versions of their program without distributing copies of their original source code.

Run-time compiled languages are slightly more flexible and informal. The program's author delivers the source code directly to the end user. Before the program can be used, it must be fed into a local compiler, which compiles and runs it. This is often done automatically for the user. One of the disadvantages of this system is that the end user must own a copy of the compiler.

On the other hand, the user who owns a copy of the compiler has the opportunity to modify the source code that was provided, or can use the code to write similar programs. The danger in modifying source code written by other authors is that the author may release an updated version of the software, and the changes that a user makes to the older version will be lost in upgrading to the new version.

Perl programs (also called Perl scripts) use CGI files over the Web, and are compiled at run-time. Although the previous discussion makes it seem that a Web user might need to

have a Perl compiler installed on the client machine to run a Perl script, that is not the case. The Web user is not actually running the CGI script; rather, a request is sent to the Web server to run the script. Therefore, it is the Web server that must have the Perl compiler installed.

2.3 Writing the Code

Although there are publicly available search engines written in Perl, the ITS staff found that it was more efficient to write a tailored search engine from scratch, instead of attempting to modify one of the existing ones.

The obvious goal of a search engine is to search Web pages. That would seem to involve calling up every Web page in FED-STD-1037C and searching it. In practice, there is much HTML coding within Web pages that does not need to be searched and that may even return incorrect results if it is searched. For example, the coding that indicates the title of the Web page would look like "`<TITLE> channel noise level </TITLE>`". If a user searches the Web pages for the word *title*, every single Web page would be returned as a positive match, since that word appears as HTML encoding in each page. The FED-STD-1037C search engine circumvents this problem with code specifically tailored to ignore several similarly irrelevant words such as the word *title*.

Another problem that ITS staff faced was that there are 5800 Web pages within FED-STD-1037C that would need to be searched. For small files like those in FED-STD-1037C, it can take approximately ten times longer to locate, open, and close a file than to extract the contents of the file. During initial tests of the search engine, it became obvious that searching 5800 Web files, one at a time, would be several orders of magnitude too slow.

To address the problem of this delay, all 5800 files were concatenated into one large file. All HTML codes were removed and "locator tags" were inserted into the file. Locator tags are ASCII codes that provide meaningful information to the search engine. For example, one tag indicates the start of the definition and another indicates its end. Another tag indicates the title of the definition and a final one indicates the file name of the definition.

By using these tags, the search engine is able to call up and to search the one large database file and to extract from it the tags representing the titles of matching definitions and the file names of those definitions. These tags can then be used to create links in a page presented to the user. The links in the "results" web page connect the user to the selected definition page within FED-STD-1037C.

The code for the initial version of the search engine provided a rudimentary list of results. In that version, the matching definitions were listed simply in alphabetical order, which was a barely adequate summary of the results. A better, sorted and ranked listing was desired.

The next version of the search engine focused on creating the weighting system described in Section 2.1.2. Many helpful enhancements were added to the layout, such as the facility for doing a second search and for displaying statistics associated with the search.

2.4 Testing the Program

Several months of testing resulted in more improvements in the basic programming of the code. For example, the Perl programming language tends to recognize the character "." (period), within a search string, as a representation of "any character." That means that if the user typed in the search phrase "r.d", the search engine would return "red", "rad", and "rod" as matching phrases. This problem was solved by having the Perl script search for and replace any periods within the search phrase with the literal representation of a period.

After incorporating the above refinements into the Web search engine, the authors tested it thoroughly, then developed a nearly identical search engine for the CD-ROM version of the glossary. Development of the CD-ROM's search engine is discussed in Section 4 below.

The search engine algorithm is discussed in the next section. Details of the search engine code are given in the appendix.

3. THE PROGRAM

This section discusses the functions and logic flow of the search engine program, the actions executed by the search engine's computer-program code, and the design of the user-interfaces used in the search engine. The appendix below contains the actual computer-program code.

3.1 The Core Functions

The search engine performs five fundamental functions:

- ✔ Examine the character strings in the entire collection of (concatenated) FED-STD-1037C Web pages to see if there are any matches to the user-entered character string.
- ✔ Assign a value to each match found (the value or "weights" are given in Table 1).
- ✔ Print out the list of matches. (The Top 25 weighted matches are listed unless the user specifies the "Top 100" or "All.").
- ✔ Print a count of the number of matches found.
- ✔ Provide a Graphical User Interface (GUI) box for users to use if another search is desired.

3.2 The Steps of the Core Functions

To perform these core functions, the search engine executes the following 24 steps:

1. The location of the supporting Perl binary files is passed to the Perl language processor.
2. The initial variables for file locations are set. These indicate to the program where to find the internal database files and how to provide the correct base HTML link for creating hyperlinks.
3. The basic weighting factors (described in Table 1) are set to determine the strength of a character string's match. [See Section 2.1.2 for complete details of the weighting factors.]
4. Several lines of code begin creating the HTML page that results from a search,

using basic HTML codes that are required at the start of all HTML pages.

5. The next lines of code start an internal timer. When the search is complete, the current time will be compared with this starting value to determine how long the search took. The next line of code following this batch sets the number of matching terms to zero. As each new match is found, this value is incremented by one.
6. The next lines of code cause the program to read in the command-line variables. These command-line variables were created by the Web page that called this Perl script. Included next in the code are two lines that are commented out (by placing the "#" sign in front of them) and these lines were used only during testing and design of the program to print out each variable as it was read into the program.
7. The next batch of code analyzes the command-line variables and sets defaults for those variables where no valid input was given by the user. If the user did not type a search phrase (a character string), the first set of lines will insert the word "space" as the search phrase. The second set of code lines will set the number of items to display equal to "25" if the user has not specified a number.
8. The next set of lines of code breaks up a multi-word search phrase into separate words. This batch of code will also recognize that some search phrases are inside quotation marks and will treat those phrases as single words. These lines of code also "sanitize" the search words by eliminating any symbols that will create incorrect search results. The period is one such character that will create incorrect results, since Perl sees a period as a wild-card indicator (*i.e.*, a period means "any one character"). Also, all lowercase letters are converted to uppercase letters. Later, during the actual search, the case of the letters is ignored.
9. The next lines of code start the actual search algorithm. This code calls up each word in the search list typed in by the user and starts the main search loop for each word.
10. For each word typed into the search by the user, the database of terms is accessed from a file on the hard disk. Each line of the database is read in, one at a time.
11. A line with just "@@" indicates that the text of the current definition is complete and that the text of the next definition will begin with the next line in the file. If this is the case, the first line of the next definition is read in, and the term name is extracted from the line of text.
12. The next set of code lines instructs the computer that, for each line that has been read into the Perl script, the computer must search for each of the target words that the user entered. The script sets the initial "best match" variable (the weighting value) to 0.25.
13. If the target word is found in the first line of the text of the definition, the "best match" variable is increased by the weighting variable (weight1) that represents a simple match. If the target word starts the line of text (*i.e.*, the target word is the term name), the appropriate weighting factor (weight2) is added to the "best match" variable. Similarly, if the

target word is found as a whole string, separated by spaces, the appropriate weighting variable (weight3) is added to the "best match" variable.

14. The target word is next sought within each additional line in the text of the definition. If it is found, a percentage of the final weighting factor (weight4) is added. This percentage is determined by calculating how close to the beginning of the line of text the match occurs. Only the first four matches within the text of the definition are considered.
15. If the user has entered multiple words or phrases to search for, and if several of them are found within one definition, an extra weight is added (weight5) to the "best match" variable. This extra weight is multiplied by a factor determined by how often each word is found (to the limit of four).
16. The next lines of code end the search loop. The loop is executed once for each definition in the database.
17. The next lines of code analyze the scores accumulated by each definition. The definitions that score the highest weighted values will rise to the top of the list. The first few lines calculate the amount of time it took to search the database.
18. The next lines calculate and print the number of definitions that contained any of the matching phrases.
19. The next lines of code separate the hits by the different target search words. The number of hits for each word is printed on the Web "results" page.
20. If the search returns no matches at all, the next few lines of code print out a simple list of possible reasons why no matching items were found. The reasons range from simple typographical errors to the absence of the typed words anywhere within the text of the definitions.
21. The next lines of code do the actual sorting of the matches to present the highest weighted matches first and the matches with the lowest weights at the bottom of the list.
22. The next lines of code provide a color background appropriate to the weighting of the individual term name listed in the results. Those matching definitions that score above 66.6% will be displayed in pink. Those definitions that score between 33.3% and 66.6% will be shown in light green, and those definitions that score below 33.3% will be shown with a yellow background.
23. The next batch of code provides the wrap-up text for the HTML Web page. These lines of code provide a blank for the user to begin another search and then the lines of code provide the final closing lines that are required on all Web pages.
24. The next lines of code constitute a subroutine, written by ITS staff, that adds the name of a definition (if any match is found) to the results "stack" of definitions that include matches. All of the associated data (*i.e.,* the "best match" value) is pushed into a parallel stack.

3.3 Displaying the Results

The list of results (printed with respective colored backgrounds to reflect the weight of each term name in the list) is displayed on the screen that announces

- ✔ all the results in weighted order
- ✔ the total number of matches ("hits")

✔ the time required to do the search.

All of the matches are listed in the scrollable, right-hand frame of the computer screen. All matches carry the underline indicating that all are hyperlinked to their respective definitions.

At the conclusion of the search, the user is invited to conduct another search, if desired.

4. CONVERSION TO CD ROM

4.1 The Format and Style

One of the primary requirements for the CD-ROM version of the search engine was that it must look and function as much like the Web version as possible. To address this requirement, the search engine was first designed and tested to function correctly over the Web. When the Web version was complete, it was copied and modified slightly for use on CD ROMs.

4.2 Supported Platforms

The Web version of the search engine can be accessed by users of most operating systems, as long as they have Internet browsing software and access to the Internet itself. The CD ROM version, however, is designed primarily for PC users of Windows 95®, 98®, and NT operating systems. These operating systems seem to be common among the expected users of this CD ROM.

Installation instructions, tailored for these systems are included on the CD ROM in a file called “start.htm”.

Users of other operating systems, such as Apple and Unix™ systems, can still use some of the files on this CD ROM. For instance, the PDF files and HTML files can be read on any operating system that can read a CD ROM and that has an Acrobat reader (for PDF files) or a network browser (for HTML files) installed.

4.3 Programming

The Perl programming language was used to create the Web version. A Perl script (*e.g.*, the search engine) is “portable” across many different operating systems. For a Perl script to run properly from the CD ROM, the user must have a version of the Perl language installed on his or her system. (This file is included on the FED-STD-1037C CD ROM.)

Most of the programming code in the Web version of the search engine was usable by the CD-ROM version without modification. The biggest difference between using the Perl script on a UNIX™ operating system and using the same Perl script on a Windows-based system is that the files on hard disk are referenced differently. For example, a file on a UNIX™ system, would be accessed as,

/files/system/help.txt

while the same file, in an equivalent location on a Windows-based system would be addressed as,

c:\files\system\help.txt

The lines in the script that contained such file path names were changed in the CD ROM version. Other minor changes were made, but the logic and the look and feel of the CD-ROM search engine is identical to the online

search engine for FED-STD-1037C. The printed results of both versions of the search engine are identical.

5. SUMMARY

5.1 Benefits to NCS and to NS/EP

Interoperability of networks and systems is achieved in large part through use of common, standardized language. Federal Standard 1037C, *Glossary of Telecommunication Terms*, which is mandated for use by all Federal Government departments and agencies, contains the agreed-upon, standard definitions for the disciplines encompassed under the umbrella of telecommunications technology. This standard advances interoperability of telecommunications equipment and services by standardizing telecommunications terminology. Interoperability, in turn, furthers NS/EP (National Security/Emergency Preparedness) goals by ensuring uniformity and portability. The standard streamlines acquisitions for the entire Federal community and supports the fastest growing segment of our economy, *viz.*, the technology segment. On the world stage, the standardized vocabulary benefits U.S. telecommunications companies, and promotes international interworking of importance to U.S. trade and security.

5.2 Benefits of Web Availability

Federal Standards have traditionally been printed and circulated in hard copy format, with mandated 5-year revisions requiring a new printing (or new change pages) for the updated edition. Distributing Federal telecommunications standards on electronic media provides direct access to the subsections of the standards themselves and wider distribution to the intended audience, while conserving paper. On-line versions of standards can be accessed even from a remote field site, using a computer and a modem.

Federal Standard 1037C on the Web and on CD ROM represents successful applications of HTML. In their recent report, Ingram and Gray [3] discuss the details of developing the HTML coding for the standard. In several ways, the large telecommunications glossary was ideally suited for presentation in hypertext format (see Section 1.2). The most significant advantage to the hypertext format is the rapidity with which users can jump from definition to definition without having to turn a precise (and often large) number of pages to arrive at the next desired definition. Other advantages of the HTML format—in addition to the advantage of speed and ease of use—include:

- ✔ use of the hypertext index
- ✔ use of search engine possibilities
- ✔ inclusion of video, audio, and animation
- ✔ Web distribution, which allows
  - greater distribution
  - ease of logistics
  - hyperlinking
  - faster (even continuous) updating
  - less expensive maintenance.

Standards like FED-STD-1037C are most useful if universally accepted; wide distribution and ease of use strengthen and enhance acceptance of a standard. The hypertext version and Web access foster wide distribution and ease of use, thereby

strengthening the standard by promoting its acceptance and wide use.

5.3 Disadvantages of the Hypertext Web Version of the Standard

The tradeoffs of the hypertext (Web) version, as contrasted to the hard-copy edition, include [4]:

- ✔ Security—The editors must remain vigilant in protecting the integrity of the online glossary against vandalism, viruses, and malicious attacks.
- ✔ Access—The editors are able to offer the glossary only to those who have access to a computer (for the CD edition) or to a computer and the Internet (for the Web version).
- ✔ Development time—The hypertext development (and proofing) of the glossary, rapid as it was, required months of added time after the hard copy was completed.
- ✔ Costs—Computer, labor, and connection costs need to be considered.
- ✔ No quick overview—Unlike the hard copy, the Web copy cannot provide the user with a glance at a full page of printed definitions or at adjacent pages for assessment of generalities such as depth and breadth of coverage or ratio of text to figures.

While the disadvantages cannot be ignored, the advantages of the hypertext version of the standard clearly outweigh the advantages of using hard copy.

5.4 Benefits of the CD-ROM Version of the Standard

In addition to making the standard's HTML files available on the Web along with a tailored search engine, ITS also developed a CD-ROM version of the standard containing the search engine and supporting software, and the PDF files of the standard (with a publicly available PDF reader). The CD also contains a licensed copy of a browser for reading the HTML files. Also included are the word-processing files, usable by anyone with the appropriate word-processing software.

The CD ROM is useful for situations where a Web connection is either unavailable or not desirable (such as in cases where firewalls or security restrictions limit Web access). Having the entire standard on a CD provides all of the advantages of the HTML Web version of the standard, with the added advantage of not requiring a connection to the Web. The two disadvantages of the CD-ROM version, as contrasted with the Web version, are:

- ✔ The CD search engine can be read only by platforms using Windows 95®, 98®, or NT operating systems.
- ✔ The search engine on the CD (which produces results identical to the same search engine on the Web) may run more slowly when searching the HTML files on the CD than on the Web. (The speed of the CD ROM search depends greatly on the equipment used.)

5.5 Benefits of the search engine

The search engine enhances use of the standard by providing capabilities for a very thorough search of any character string within

all 5800 entries. With the search engine, users can build their own exacting index to mold the standard into a very powerful research tool. Benefits of the search engine include:

- ✔ Complete, tailored access to the text of all definitions containing the specified character string used in the search, resulting in more comprehensive use of the glossary.
- ✔ A ranking of results of the search.
- ✔ More rapid use of the entire glossary (in preparation for an updated edition).

5.6 Advancing the Purpose of the Standard

In addition to promoting interoperability and furthering NS/EP goals, the standardized vocabulary operates on the world stage to benefit U.S. telecommunications companies by promoting international interworking of importance to U.S. trade.

Telecommunications terms describing new discoveries and technological advances must be communicated quickly and accurately to prospective users if those terms are to be used effectively in technical specifications and procurement documents. Recent advances in computer security, network architecture, network security, Internet technologies, and photonics have contributed much to telecommunications technology. One of the fastest, most pervasive and far-reaching means of distributing new standardized terminology is via the Web. Distributing the 1037C glossary and its search engine on the Web enhances the use of its standardized vocabulary, promotes a common language and communicates news of the benefits of advances in telecommunications technology.

6. REFERENCES

[1] X3K5, *ANSDIT, American National Standard for Information Technology*, ANSI X3.172, 1997.

[2] ISO/IEC, ISO/IEC 2382: *Information Technology—Vocabulary*, 1999.

[3] W. Ingram and E. Gray, *A Federal Standard on Electronic Media,* NTIA Report 98-350, Aug. 1998.

[4] E. Gray and W. Ingram, "A Hypertext Glossary: More than print with bells and whistles," *Proceedings 1998 STC Region 7 Conference, The Medium Formerly Known as Print,* Oct. 15-17, 1998, Denver, Colorado.

[5] L. Wall and R. L. Schwartz, *Programming Perl*, O'Reilly and Associates, Inc., Sebastopol, California, March 1992.

APPENDIX: COMPUTER CODE FOR THE SEARCH ENGINE

This appendix presents the verbatim computer-program code developed to create the search engine for Federal Standard 1037C, *Glossary of Telecommunication Terms*. Annotations and descriptions of important program components are included.

✏ The following line indicates to the Perl language processor where to find the supporting Perl binary files.

```
#!/usr/local/bin/perl
```

✏ The following lines set the initial variables for the file locations. They indicate where to find the internal database files and how to provide the correct base HTML link for creating hyperlinks.

```
# -- Setting base variables
$base = "http://www.its.bldrdoc.gov/fs-1037";
$dir = "/nd/bing/public_html/1037e";
$nbase = "/nd/bing/public_html/pub";
$hbase = "http://www.its.bldrdoc.gov/fs-1037";
```

✏ The following lines define the basic weighting factors to determine the strength of a phrase match. See Section 2.1.2 for complete details of the weighting factors.

```
$weight1 = 10; # weighting factor if phrase in title
$weight2 = 30; # weighting factor if phrase IS title
$weight3 =  5; # weighting factor if phrase is whole word
$weight4 =  2; # weighting factor if phrase is in a definition
$weight5 = 10; # weighting factor if all phrases found
$maxhits =  4;
```

✏ The following lines begin creating the HTML page that results from a search. The lines are the basic HTML codes that are required at the start of all HTML pages.

```
# -- Print out the basic header info
print "Content-type: text/html\n\n";
print "<BODY background=\"$base/gifs/\$-backg2.gif\" text=\"#000000\"
bgcolor=\"#ffffff\" link=\"#FF0000\" vlink=\"#0000ff\" alink=\"#00FF00\">\n";
print "<Style> A.HL0 {background-color: white; color: silver}</Style>\n";
print "<Style> A.HL1 {background-color: #ffffcc; color: blue}</Style>\n";
print "<Style> A.HL2 {background-color: lightgreen; color: green}</Style>\n";
print "<Style> A.HL3 {background-color: pink; color: red}</Style>\n";
print "<html><title>Search Results</title>\n";
print "<h1><center>Search Results</center></h1>\n";
```

✏ The following lines start the internal timer. When the search is complete, the current time will be compared with this starting value to determine how long the search took. The next line

sets the number of matching terms to zero. As each new match is found, this value will be increased by one.

```
# -- Start the timer
$tstart = time;
$nhits = 0;
```

✏ These lines cause the program to read in the command-line variables, which were created by the Web page that called this Perl script. The two following lines are "commented out" (by placing the "#" sign in front of them) and were used to print out each variable on the monitor, as it was read into the program, during testing and design of the program.

```
# -- Read in the variables from the command line from Web page
&ReadParse;
 #foreach $key (keys %in) {
 #print "$key: $in{$key}<br>\n"; }
```

✏ The following lines analyze the command-line variables and set defaults for those variables where no valid input was given by the user. If the user did not type a search phrase, the first set of lines will insert the word "space" as the search phrase. The second set will set the number of items to display to 25 if the user has not specified a number. The third set of lines will set the default document to search to be Federal Standard 1037C if none is specified. This option is commented out, since, at this time, Federal Standard 1037C is the only on-line Federal Standard that can be searched with this program.

```
# -- Print out warning messages and set default values
 print "<ul>\n";
 if ($in{searchphrase} eq "") {
  print "<li>You didn't type anything into the search blank, so I did a search
for '<i>space</i>' (note the irony).<br>\n";
  $in{searchphrase} = "space";
 }

 if ($in{just25} eq "") {
  print "<li>You didn't select how many results to display, so I limited the
display to the <i>top 25</i> matches.<br>\n";
  $in{just25} = "25"
 }

 # if ($in{tosearch} eq "") {
 # print "<li>You didn't select a document to search, so I searched <i>FS-
1037C</i>.<br>\n";
 # $in{tosearch} = "37"
 # }
```

✏ The next lines will break up a multi-word search phrase into separate words. They will also recognize that some search phrases are inside quotation marks and will treat those phrases as single words. These lines of code also try to "sanitize" the search words by eliminating any symbols that will create incorrect search results. The period is one such character that will create incorrect results, since Perl sees a period as a wild-card indicator (*i.e.*, a period means

"any one character"). Also, all lowercase letters are converted to uppercase letters. Later, during the actual search, the case of the letters is ignored.

```
# -- Break up searchphrase
 $oldphrase = $in{searchphrase};
 $in{searchphrase} =~ s/([^a-zA-Z0-9_ \r\t\n\f"?*])/\\$1/g;
#print "$in{searchphrase}\n"; die;

 $xsearch = $in{searchphrase};
while ($xsearch =~ /\"/) {
 $ysearch = $xsearch;
 $xsearch =~ s/(.*)\"(.*)\"(.*)/$1$3/;
 $ysearch =~ s/(.*)\"(.*)\"(.*)/$2/;
 #print "$ysearch\n";
 push (@wordlist,$ysearch);
 }

 $xsearch =~ s/,//g;
 $xsearch =~ s/^\s*//g;
 $xsearch =~ s/\s+/ /g;
 $xsearch =~ s/\s*$//g;
@word2 = split (/ /,$xsearch);
foreach $i (@word2) {
 #print "$i\n";
 if ($i eq "and" || $i eq "or") {} else {
  push (@wordlist,$i)
  }
 }
 # push (@wordlist,@word2);
```

✏ These lines are the start of the actual search algorithm. They call up each word in the search list and start the main search loop for each one.

```
foreach $i (@wordlist) {
 $wordhits{$i} = 0;
 }
```

✏ For each word, the database of terms is accessed from a file on the hard disk. Each line of the database is read in, one at a time.

```
# -- Open Database File and read each definition
open (FSDB, "$nbase/fsdb.txt");
 while ($line = <FSDB>) {
  chop($line);
```

✏ A line containing just "@@" indicates that the text of the current definition is complete and that the text of the next definition will begin with the next line in the file. If this is the case, the first line of the next definition is read in, and the term name is extracted from the line of text.

```
   # -- For each file, set defaults
   if ($line eq "@@") { # Indicates next definition
           # and begin special processing.
```

```
&push_stack;
$href = <FSDB>;
 chop($href);
$fname = $href;
 $fname =~ s/.*(dir-...\/_....\.htm).*/$1/;
$def = $href;
 $def =~ s/.*main1">(.*)<\/a>/$1/;
```

✏ The next lines instruct the computer to search for each of the target words that the user entered for each line that has been read into the Perl script. The script sets the initial "best match" variable to 0.25.

```
 # -- Try each phrase
 foreach $searchphrase (@wordlist) {
  $hit{$searchphrase} = 0.25;
```

✏ If the target word is found in the first line of the text of the definition, the "best match" variable is increased by the weighting variable (weight1) that represents a simple match. If the target word starts the line of text (*i.e.*, the target word is the term name), the appropriate weighting factor (weight2) is added to the "best match" variable. Similarly, if the target word is found as a whole string, separated by spaces, the appropriate weighting variable (weight3) is added to the "best match" variable.

```
 # -- Add extra if hit is in Def.
 if ($def =~ /$searchphrase/i ) {
  $besthit = $besthit + $weight1;
  if ($def =~ /^$searchphrase$/i ) { $besthit = $besthit + $weight2 }
  if ($def =~ /\W$searchphrase\W/i ) { $besthit = $besthit + $weight3 }
  if ($thisword{$searchphrase} == 0) {
   ++$thisword{$searchphrase};
   ++$hit{$searchphrase}
   }
  } #End if in Def name
  } #End each phrase
```

✏ The target word is sought within each additional line in the text of the definition. If it is found, a percentage of the final weighting factor (weight4) is added. This percentage is determined by calculating how close to the beginning of the line of text the match occurs. Only the first four matches within the text of the definition are considered.

```
 } else {
 # If not new def, do normal processing

 # -- Try each phrase
 foreach $searchphrase (@wordlist) {

# -- and then examine each line
 $lbefore = length($line); # Determine line length
  #print "$line\n";
 $liney = $line;
 while ($liney =~ /$searchphrase/i) {
  if ($thisword{$searchphrase} == 0) {
```

```
      ++$thisword{$searchphrase};
      ++$hit{$searchphrase};
      }
     if ($liney =~ /\W$searchphrase\W/i) {
      $liney =~ s/(.*)$searchphrase.*/$1/i;
      $lafter = length($liney); # Determine chopped line length
      $besthit = $besthit + $weight3 * (1 - ($lafter/$lbefore));
      } else {
      $liney =~ s/(.*)$searchphrase.*/$1/i;
      $lafter = length($liney); # Determine chopped line length
      $besthit = $besthit + $weight4 * (1 - ($lafter/$lbefore));
      }
     } # End 'found phrase' While loop
     } # End for each phrase
```

✏ If the user has entered multiple words or phrases to search for, and if several of them are found within one definition, an extra weight is added (weight5) to the "best match" variable. This extra weight is multiplied by a factor determined by how often each word is found (to the limit of four).

```
     ## -- Calculate weight5
     $j = $weight5 * $#wordlist; $j2 = $j;
      #print "$j\n"; die;
     foreach $i (@wordlist) {
      if ($hit{$i} > $maxsize) {$hit{$i} = $maxsize}
      $j = $j * $hit{$i}; $j2 = $j2 *.25;
      }
      if ($j != $j2) {$besthit = $besthit + $j}
```

✏ These next lines are the end of the search loop. The loop is executed once for each definition in the database.

```
   } # End if @@
  } # end while
  &push_stack;

#$nword = $in{word};
#print "Best Score: $verybest<br>\n";
```

✏ These lines then analyze the scores accumulated by each definition. The definitions that score the highest weighted values will rise to the top of the list. The first few lines calculate the amount of time it took to search the database.

```
# -- Sort and print the files by most likely hit
$tend = time;
$tsecs = $tend - $tstart;
print "<li><font color = red>5860</font> definitions searched in <font color =
blue>$tsecs</font> seconds (actual time to transmit results to your browser
may have taken longer)\n";
```

✏ These lines calculate and display the number of definitions that contained any of the matching phrases.

```
print "<li><font color = red>$nhits</font> definition";
 if ($nhits != 1) {print "s"}
print " found that contain the item";
 if ($#wordlist > 0) {print "s"}
print " ";
```

✏ These lines separate the hits by the different target search words. The number of hits for each word is displayed on the Web page.

```
$i = -1;
 foreach $word (@wordlist) {
 $wh = $wordhits{$word};
 $word =~ s/\\([^\\])/$1/g;
 print "\"<font color = green><b>$word</b></font>\"";
  if ($#wordlist > 0) { print " (<font color = red>$wh</font>)" }
 ++$i;
 if ($i == $#wordlist) {
  print ".\n</ul><hr>\n";
  } else {
   if ($i == ($#wordlist - 1)) {print " and/or "} else {print ", "}
  }
 }
```

✏ If the search returns no matches at all, these next few lines display a simple list of possible reasons why no matching items were found. The reasons range from simple typographical errors to the absence of the typed words anywhere within the text of the definitions.

```
 if ($nhits == 0) {print "<ul><li>Here are some possible <a href=\"$hbase/fs-
0.htm\" target=\"main1\">reasons</a> why you found no matches.</ul>\n"}
```

✏ These lines sort the matches to present the highest weighted matches first, followed by matches with lower weights.

```
$i = 0;
foreach $fname (sort { $hitsize{$b} <=> $hitsize{$a} } keys %hitsize) {
 #print "$fname, $hitsize{$fname}, $hitname{$fname}\n";
 #$perc = int(10000*$hitsize{$fname}/$verybest)/100;
 ++$i; if ($i > $in{just25}) {last}
 $perc = int(100*$hitsize{$fname}/$verybest+.5);
```

✏ These lines will provide a color background appropriate to the weighting of the individual term name listed in the results. Those matching definitions that score above 66.6% will be displayed in pink, those definitions that score between 33.3% and 66.6% will be shown in light green and those definitions that score below 33.3% will be shown in yellow.

```
    $highl = 0;
     if ($perc >= 0) { $highl = 1 }
     if ($perc > 33.33) { $highl = 2 }
     if ($perc > 66.67) { $highl = 3 }

 print "<i>$perc%</i> -- <a class=HL$highl
$hitname{$fname}</a></style><br>\n";
 }
```

✏ These lines provide the "wrap up" text for the HTML Web page. They provide a blank for the user to begin another search and then they provide the final closing lines that are required in all Web pages.

```
print "<hr>Try Another Quick Search!<br>\n";
print "<FORM METHOD=\"POST\" ACTION=\"http://132.163.64.201/cgi-
bin/searchfs.prl\" TARGET=\"main2\">\n";
print "Phrase to Search for:<br><INPUT NAME=\"searchphrase\" SIZE=25><br>\n";
print "<br>How many matches to show:";
print "<br><INPUT TYPE=\"radio\" NAME=\"just25\" VALUE=\"25\">Top 25";
print "<INPUT TYPE=\"radio\" NAME=\"just25\" VALUE=\"100\">100";
print "<INPUT TYPE=\"radio\" NAME=\"just25\" VALUE=\"6000\">All";
print "</form>";
print "</html>\n";
```

✏ This subroutine, written by ITS staff, adds the name of a definition (if any match is found) to the results "stack" of definitions that include matches. All of the associated data (*i.e.*, the "best match" value) is pushed onto a parallel stack.

```
# ------------------------------------------------
# -- Push hits onto stack subroutine
    sub push_stack {
    foreach $i (@wordlist) {
     if ($thisword{$i} > 0) {++$wordhits{$i};
     #print "<br>WHIT($wordhits{$i}) -
$liney<br>\n$besthit<br>\n$wordlist{$i}<br>\n";
      $thisword{$i} = 0; }
     }
     if ($besthit > 0 ) {
      #print "$liney\n$def\n$fname\n$besthit\n$wordlist{$searchphrase}\n";
die;
      if ($besthit > $verybest) { $verybest = $besthit }

      # print "$fname, $i -- ";
      ++$nhits; # Count the number of hit files
      #print "<br>NHIT($nhits) -
$liney<br>\n$besthit<br>\n$wordlist{$i}<br>\n";
       $hitsize{$fname} = $besthit;
       $hitname{$fname} = $href;
      } # End if
     $besthit = 0;
     } # End stack push
```

✏ The following subroutine[4] reads command-line variables generated by the search-engine Web page and based on the text that the user types into the blanks on the Web page. In this instance, the variables are the target search phrase and the number of matching entries to return.

[4] Permission to use this subroutine was granted by the owners of the copyright, via the Web page where this, and other useful Perl subroutines, are made available for Perl programmers. That page is, *"The cgi-lib.pl Home Page,"* and is located at URL, http://cgi-lib.stanford.edu/cgi-lib/. This Web page was last accessed on March 5, 1999.

```
# --------------------
# Adapted from cgi-lib.pl by S.E.Brenner@bioc.cam.ac.uk
           # Copyright 1994 Steven E. Brenner
           sub ReadParse {
             local (*in) = @_ if @_;
             local ($i, $key, $val);

             if ( $ENV{'REQUEST_METHOD'} eq "GET" ) {
               $in = $ENV{'QUERY_STRING'};
             } elsif ($ENV{'REQUEST_METHOD'} eq "POST") {
               read(STDIN,$in,$ENV{'CONTENT_LENGTH'});
             } else {
                   # Added for command line debugging
                   # Supply name/value form data as a command line argument
                   # Format: name1=value1\&name2=value2\&...
                   # (need to escape & for shell)
                   # Find the first argument that's not a switch (-)
                   $in = ( grep( !/^-/, @ARGV )) [0];
                   $in =~ s/\\&/&/g;
             }

             @in = split(/&/,$in);

             foreach $i (0 .. $#in) {
               # Convert plus's to spaces
               $in[$i] =~ s/\+/ /g;

               # Split into key and value.
               ($key, $val) = split(/=/,$in[$i],2); # splits on the first =.

               # Convert %XX from hex numbers to alphanumeric
               $key =~ s/%(..)/pack("c",hex($1))/ge;
               $val =~ s/%(..)/pack("c",hex($1))/ge;

               # Associate key and value. \0 is the multiple separator
               $in{$key} .= "\0" if (defined($in{$key}));
               $in{$key} .= $val;
             }
             return length($in);
           }
```

This program was written by ITS staff under the direction and sponsorship of NCS.