DReSS 2.0: Lightweight Groupware for Hypertext Publishing on the Web

Ad Aerts, Paul De Bra* and Marco Timmermans
Information Systems Section, Department of Computing Science
Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, The Netherlands
E-mail : {wsinatma, debra, marcot}@win.tue.nl

Abstract: We present recent evolutions in the development of *DReSS*, a "*D*ocument *Re*pository *S*ervice *S*tation". DReSS allows World Wide Web users to collaboratively create and maintain hypermedia documents on a Web server without the need for a login (or ftp upload) access to the server machine. DReSS emphasizes the manipulation of hyperdocuments consisting of several files that are linked together. DReSS is a small system, that is easy to install on any Web server, and that offers users the ability to up- and download sets of files. At the Eindhoven University of Technology DReSS has recently been integrated with an adaptive courseware system to enable students to upload assignment work to the Web-server containing the course text and the assignments. New features in DReSS 2.0 include the ability to use a browser's upload capabilities, a new and more user-friendly user-interface, the ability to use a browser's upload facilities, and machine independent client helper applications and server-side scripts, all written in Java.

1. Introduction

World Wide Web (or WWW) [Berners-Lee 94] is a distributed publishing platform. Authors create and change documents locally and upload them to a server, where they can be accessed by their collaborators. While downloading (or "reading") documents is usually allowed for everyone, and done in an anonymous way, authors need login or ftp upload permissions on the server machine in order to publish their documents. Alternatively, in a local area network (Intranet) the server may have access to the author's machine or home directory, and copy or even serve the documents from there. To avoid the need for such permissions some document repository systems have been created like BSCW [Bentley 95] and DReSS [De Bra 95].

Groupware is defined [Bock 95] as a computer based system that supports people in the execution of common tasks (or the achievement of a common goal) and that provides an interface to a shared environment. Document repository systems represent a lightweight form of groupware: they offer uploading and sharing of documents among varying groups of authors. Through Internet technology we can create shared workspaces that can be reached from outside the institute or company (see also [Goossenaerts 97] for a business oriented application of a shared workspace) and support collaboration in geographically distributed environments. Apart from shared workspaces systems like BSCW also offer (threaded) discussion groups and on-line meetings.

DReSS (for *D*ocument *Re*pository *S*ervice *S*tation) has been developed at the Eindhoven University of Technology. It is a tool that provides collaborative information sharing; it enables authors to move hyperdocuments to the WWW server and to update such documents later on. The emphasis of DReSS is on the support for *hypertext* documents, i.e. for documents that consist of a possibly large number of small files that are connected by hypertext links. BSCW (Basic Support for Collaborative Work) is sometimes compared to DReSS because both systems offer shared workspaces. However, DReSS is specifically designed to be *just* a document repository, while BSCW aims at providing more complete support for collaborative work, and therefore includes many features not found in DReSS. The strong points of DReSS are the transparent use of URLs, and the ability to upload sets of files in one operation. BSCW is only able to resolve hypertext links (using the HTML <A> tag) in some limited cases, while DReSS handles links transparently, even between documents in different workspaces. At the Eindhoven University of Technology we have integrated DReSS into an adaptive courseware system [De Bra 96, Calvi 97] in order to enable students to upload assignment work to the Web-server containing the course text and the assignments. Such assignment work typically consists of sets of pages

---

* Paul De Bra is also affiliated with the "Centrum voor Wiskunde en Informatica" in Amsterdam and the University of Antwerp, Belgium.

that contain links to each other and that also contain images, applets, and possibly other types of objects as well. It is important for students to be able to upload assignment work in a single operation, and perform validation tests on the uploaded work immediately.

2. Document Repository Systems

2.1 Requirements for and History of DReSS

The main reason for using a document repository system is to avoid the need for authors to have login or ftp upload permission on the server machine, or for shared file systems between clients and servers. More advanced features like link databases, versioning, and on-the-fly generation of indexing information are worthwhile, but secondary to the goal of providing a document repository system that is easy to install and use, and that can be used with different browsers (and servers) without requiring any modifications to "standard" WWW software. In 1995 *DReSS*, a tool that turns a WWW server into a "*D*ocument *Re*pository *S*ervice *S*tation", was introduced (version 1.1 was described in [De Bra 95]). DReSS enables authors to move documents to the WWW server, and to update documents on the server, without compromising the server's or client's security. To ensure that the security of the server and the author machines is not compromised by using DReSS, the set of requirements given below contains several constraints related to security:

- Authors need no login permission on the server machine.
- The server machine needs no (NFS) access to the author's home directory (or any other directory on the author's machine).
- The author's machine must not need to host an ftp- or WWW-server in order to upload documents to the main server.
- All communication is done by means of HTTP. (This may be changed to S-HTTP or HTTP over SSL at a later date for better security.)
- The system must use standard and/or platform independent software: unmodified WWW browsers and servers, and only Java programs where additional software is needed.
- Most of the functionality of the system must be implemented on the server machine (so as to use "thin" clients).
- The author's WWW browser is the user-interface for the whole system.
- Elaborate access control is needed. The Webmaster must be able to decide which authors may create and edit documents in which directories and the authors must be able to decide who may edit and view their documents.
- The system must prevent simultaneous editing of the same document by different authors.

DReSS, version 1.1 met most of the above requirements. It used CGI-scripts on the server side, and two small auxiliary programs on the client side that were completely invisible to the author. These programs were based on the common WWW library (from W3C) to ensure easy portability. However, that WWW library proved to be difficult to port to non-Unix systems such as Windows-3.1 and Windows-95. Version 1.1 of DReSS was written in C and used on Solaris (release 2.4 on both Sparc and x86) and SunOS 4.1.3 for both the client and server, although client and server could run on a different OS.

Several changes in the IT-environment have taken place since the introduction of DReSS in 1995. A standard for file uploading was defined (cf. RFC 1867 [Nebel 95] which is supported by such browsers as Netscape Navigator. Office environments like the MS-Office and the Wordperfect Office suite offer built-in support for HTML-editing. The Java programming language has been introduced and has found widespread support and usage for Internet/Intranet applications. These developments have led us to reconsider the design and implementation of DReSS to take advantage of these developments. This led to the present version of DReSS: version 2.0. Our key design goal has been to build a low key application that requires a minimum of changes to the Web-client and -server. To make version 2.0 of DReSS portable, both the client helpers and server scripts are written in Java. For DReSS 1.1 two client helper programs were always required. In DReSS 2.0 these helpers are only required for handling hyperdocuments that consist of several files. Single file upload and download can be handled by any modern browser (which supports RFC 1867). The two helper programs perform the following functions:

- When a document is retrieved from the server, a helper program can be used to start a local editor (instead of just saving the document on disk).

- When the author has finished editing the document a second helper program uploads the document to the server. (The browser can do this as well, for a single file.) When a set of documents is uploaded to the server, the transfer is performed using a single HTTP request. (In DReSS 1.1 uploading a set of documents required a request per file.)

2.2 Client- and Server-Side Helper Applications

The architecture of DReSS (and also of the document repository part of BSCW) consists of four standard components and two types of helpers. Figure 1 shows this architecture.

Figure 1: DReSS architecture

- On the client side the author of a document is using a standard editor (for text, images or other data types) to create and modify documents, and a browser to interact with the repository system on the server.
- A client-helper application guides the communication between the editor(s) and the browser. It ensures that the editor receives as input the document which the browser received earlier from the server. It also ensures that when the server sends an archive containing a set of documents the individual files are extracted from the archive before an editor is invoked. Another helper ensures that the document is sent back with the appropriate name, and also takes care of uploading sets of documents.
- On the server side a database (called "repository") maintains not only the documents but also some meta-information and history of each document.
- A server-side helper, in the case of DReSS a CGI-script, handles the communication between the Web server and the repository system. It generates appropriate confirmation or error messages for each request from the browser, and it archives sets of documents before sending them to the browser and extracts them from archives before depositing them back in the repository system.

The basis for activating the correct helper program, and for using the correct archival formats is the use of MIME-types. (MIME stands for Multi-purpose Internet Mail Extensions.) A WWW-browser has, as part of its configuration, a list of MIME-type/application pairs. When a Web-server answers a request it sends an HTTP-reply to the server. This reply consists of a standard header, possibly followed by a document. The Content-Type field of that header defines the MIME type of the document that follows. The browser checks its list to determine whether it should try to display the document all by itself, whether to use a plug-in the user has installed, or whether to use a separate (external) application. In order to use DReSS, a special new MIME type needs to be added to the browser's list, and bound to the appropriate helper application. (On a Unix system this binding can be done in the user's .mailcap file, thus without changing any of the browser's settings.)

Up- and downloading multiple files is achieved through the use of multipart MIME types (as defined by RFC 1521). This eliminates the need for special archival programs on the server- and client-side, of which few are readily available on all platforms. The DReSS helpers take care of bundling and extracting files to and from the multipart messages.

The possibilities of file systems for maintaining status information about documents are insufficient for document repository systems. DReSS maintains the following information about a document:

- **creator**: the author who first created this document; this info cannot be changed.
- **current author**: the author who is "currently" editing the document.
- **lock status**: when an author wishes to modify a document, that document is "locked" and the author becomes the "current author"; only the current author can release the lock.
- **authors**: the authors who are entitled to modify this document; this is like the Unix "group" concept, but each document by have a different set of authors, without the need for explicitly naming groups beforehand.
- **log file**: a log of all actions (downloads and uploads) performed on the document.

Unlike BSCW DReSS does not have a versioning system, which provides access to all versions of a document that ever existed. Documents are stored as files on the Web server and are available through the URL, which corresponds to that file. This makes it easy to use hypertext links between documents in the same and in different workspaces. The files on the server are all owned by the WWW server daemon. Ownership information is kept in the status database.

3. User-Interface for DReSS

While working with DReSS version 1.1 we identified several points that needed attention to improve the usability of DReSS:

- DReSS 1.1 only supported one editor (the default was Emacs, but this could easily be changed). DReSS 2.0 supports more editors and a dialog for configuring the editor from within the browser. For different media types such as text, and images, different editors are needed, but also for different textual document formats such as HTML, and MS-Word.
- The original dialog between the user and DReSS was too complicated. The user-interface almost always displayed many buttons for actions that were not possible in the current state. Figure 2 and Table 1 show the (simplified) state-transition diagram for DReSS 2.0.
- The upload protocol supported by DReSS 1.1 was implemented using an ad-hoc encoding, thus preventing the use of the emerging browser-capability to upload files according to RFC 1867. DReSS 2.0 uses the protocol of this RFC, and allows single file upload from within the browser. (Multiple file upload is still handled through a helper application.)
- The multi-file upload in DReSS 1.1 was limited to uploading an entire directory. In DReSS 2.0 uploading specific sets of files can be done through the client uploader interface, which is shown in Figure 3.

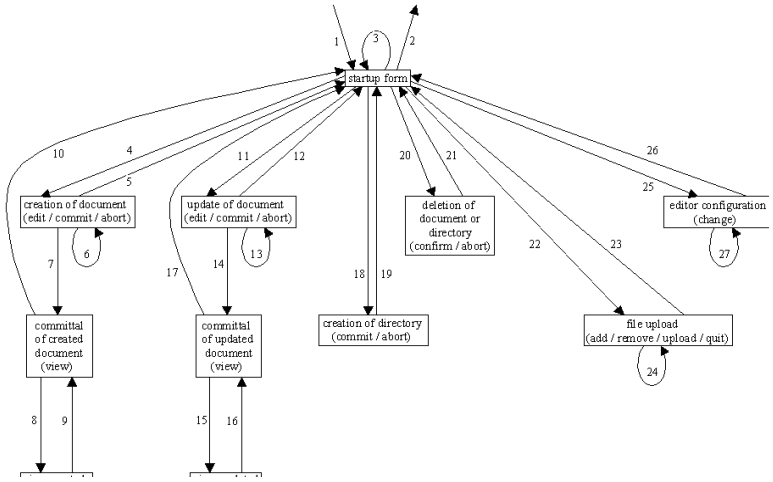

Figure 2: State transition diagram for DReSS 2.0.

1. start dress
2. “back” / “home” (quit dress)
3. clear fields / create document (already existing file or directory) / update document (non-existing file or directory)
4. create document
5. abort / “back”
6. edit
7. commit
8. view
9. “back”
10. “back”
11. update document
12. abort / “back”
13. edit
14. commit
15. view
16. “back”
17. “back”
18. create directory
19. confirm / abort / “back”
20. delete document or directory
21. confirm / abort / “back”
22. upload file(s)
23. upload / quit
24. add / remove
25. configure editor
26. “back”
27. change editor

Table 1: Transitions

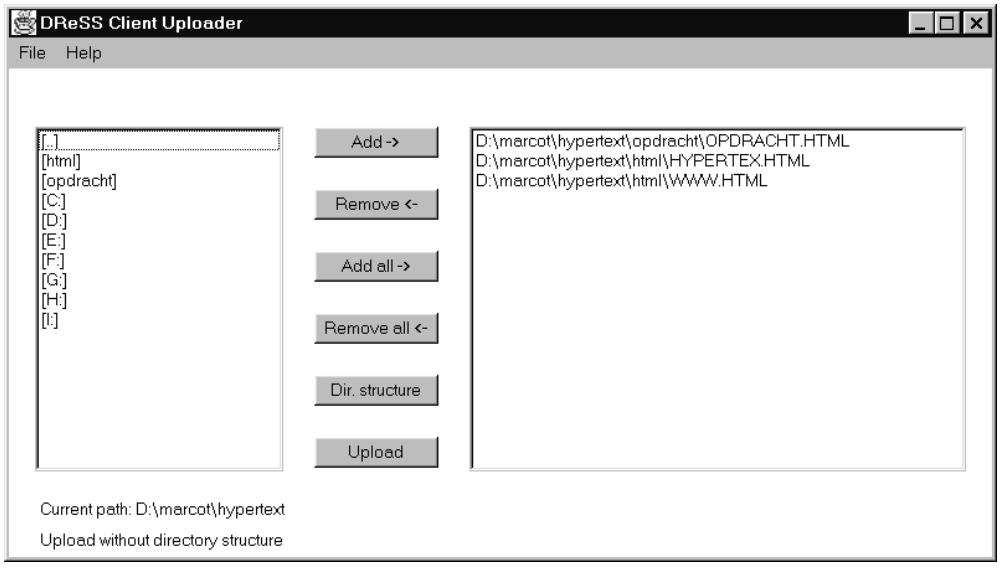

Figure 3: Client Uploader Interface

4. Application Issues for DReSS

4.1 Machine independence

The main problem we encountered in employing DReSS version 1.1 was the limited portability of the client helper programs. Despite the use of the standard WWW library from W3C it proved difficult to port the helpers to non-Unix systems. This prevented us from employing DReSS for its primary purpose: to enable students to upload assignment work for the hypermedia course 2L670 [De Bra 96]. A finished assignment is a hyperdocument of between 50 and 150 pages (files), so employing a single file upload system like BSCW was out of the question. To solve this problem, and to provide better file (or directory) browsing as well, the helper programs for DReSS 2.0 are written entirely in Java, which means that they should run unmodified on any platform that supports Java. (They have been tested on Windows'95 and Solaris, which are the platforms used by most of our students.)

In order to achieve machine independence on the server-side as well, the CGI-scripts for the document repository have been rewritten in Java. However, this does not result in a completely platform-independent solution, since CGI relies on the ability to pass environment variables from the Web server to the scripts, something which is not possible with Java programs from JDK version 1.1 onwards. (Reading environment variables has not only been marked deprecated but has been effectively removed from the Java support libraries.) The need for reading environment variables is circumvented by wrapping the CGI-scripts in a shell script. The needed shell script for Unix is different from the necessary "bat" file for Windows.

4.2 Groupware aspects

In the present configuration DReSS supports a limited form of groupware functionality, called information sharing [Gibbs 91], which is aimed at the use (and modification) of the same information by different people for performing knowledge intensive activities. BSCW, (and also Lotus Notes) offers an integrated message board, similar to HyperNews or WWWboard. For our courseware we decided to use HyperNews which was readily available, rather than developing yet another message board.

Lotus Notes supports replication of documents but does not offer an automated conflict resolution strategy for updates of the primary copy. In DReSS conflict prevention is realized by providing locks on documents currently being edited by one of the authors. Locking is done at the document level. In [Munk 96] we studied the use of an object oriented database for storing hypertext. The database replaces the file system, and the

WWW-server is modified to store documents directly in the database. By using an object model for hyperdocuments which allows them to be decomposed according to their HTML-structure it becomes possible to access paragraphs or smaller parts in a document. Using the locking mechanism of the database system a more fine grained access to documents can be supported allowing different authors to work concurrently on different parts of the same document. Integration of DReSS with the OODB-based WWW-server is a project which will be started in the near future.

Recently we started another DReSS-related project: to combine DReSS with a Workflow Engine, based on the ExSpect (Executable Specification Tool) system [Somers 93] also developed at the Eindhoven University of Technology. This effort is part of an educational project aimed at student collaboration in a Software Engineering course. DReSS is used for sharing the deliverables of the assignment work for this course between members of a group of students who are jointly performing the assignment. The workflow engine is used to guard the orderly executions of the various activities of the project life cycle, and the timely delivery of the different required documents.

5. Conclusions

DReSS (in release 2.0) has evolved to a platform independent document repository system, through the use of standard protocols (RFC 1521 and 1876) and helper applications written in Java. The main advantages of DReSS over similar systems (like BSCW) are:

- **Simplicity**: DReSS is small and easy to install on most Web servers. It offers shared workspaces, to which users can upload documents, which can then be updated by a group of authors (but one at a time).
- **Hypertext Functionality**: DReSS is especially suited for applications where documents consist of many files, possibly connected by means of hypertext links. A whole set of files can be uploaded or downloaded at once.

In the near future we will be evaluating the use of DReSS as part of our hypermedia courseware. This evaluation will be used to further improve the user-interface and the dialog between user and system.

6. References

[Bentley 95] Bentley, R., Horstmann, T., Sikkel, K., Trevor, J., Supporting collaborative information sharing with the World-Wide Web: The BSCW Shared Workspace system, *Fourth International WWW-Conference*, Boston, 1995.

[Berners-Lee 94] Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H., and Secret, A., The World-Wide Web, *Communications of the ACM*, 37(8): 76-82, 1994.

[Bock 95] Bock, G.E. and Marca, D.A., *Designing Groupware: A guidebook for designers, implementators and users*, New York, McGraw-Hill, 1995.

[DeBra 95] P.M.E. De Bra, A.T.M. Aerts, Multi-User Publishing in the Web: DReSS, A Document Repository Service Station, *NLUUG Autumn 1995 Conference on Publishing in the World Wide Web*, 1995, URL: http://wwwis.win.tue.nl/~debra/dress/dress-paper.html

[De Bra 96] De Bra, P., Teaching Hypertext and Hypermedia through the Web. In: *Proceedings of the WebNet'96 Conference*, 130-135, San Francisco, 1996.
(URL:http://wwwis.win.tue.nl/~debra/webnet96/index.html)

[Calvi 97] Calvi, L., De Bra, P., Using Dynamic Hypertext to create Multi-Purpose Textbooks. (to appear) In: *ED-MEDIA'97*, Calgary, 1997. (URL: http://wwwis.win.tue.nl/~debra/ed-media97/)

[Gibbs 91] S. Gibbs, CSCW and software engineering, in: *Groupware 91: The potential of team and organisational computing*, SERC, Amsterdam, 1991.

[Goossenaerts 97] Goossenaerts, J., Aerts, A.T.M., Filip, F., Sauer, C.: Product Configuration in the Framework of the Virtual Enterprise, In P. Groumpos (ed.) *Life Cycle Approaches to Production Systems, ASI'97*, Budapest, July 1997, 33-38.

[Munk 96] Munk, J.T. de, Aerts, A.T.M. , De Bra, P.M.E., OODB Support for WWW Applications: Disclosing the internal structure of Hyperdocuments, *World Conference of the Web Society: WebNet96*, San Francisco, 1996.
URL: http://wwwis.win.tue.nl/~debra/webnet-2/index.html

[Nebel 95] Nebel, E., and Masinter, L., Form-based File Upload in HTML, *RFC 1867*, Xerox Corporation, November 1995.

[Somers 93] L.J.Somers, M.Voorhoeve, and K.M.v.Hee, Improving software design quality with ExSpect, *Sixth International Workshop on Computer-Aided Software Engineering*, 174-177, Singapore, 1993.