Distributed Development Project using WWW

Ivica Crnković

ABB Industrial Systems AB, dept LKST

S-721 67 Västerås, Sweden

E-mail: ivica@sw.seisy.abb.se

ABSTRACT: This paper describes a support for a software development distributed geographically and used on different platforms. By using simple cgi-scripts it is possible to access the development project structure and to exchange the software components from different sites. In particular a solution for a distributed software configuration management is presented: A development structure is placed on a WWW server and under control of a configuration management. Software components are via Netscape checked-out, locked for double updating and then checked back in the project repository. The software components and their states are shown by simple graphical objects in HTML pages that are dynamically created on the WWW server site.

1. INTRODUCTION

The enormous expansion of the Internet and usage of the World Wide Web (WWW) has removed barriers between different platforms and geographical distances. However, in the same time the new requirements on software applications have been posed: applications must follow common standards, must available on any operating system, must have a possibility to access distributed data. These requirements are even more valid for applications supporting software development, they must be distributed but they also must support a development of distributed applications. A client/server concept is a common technique for this type of applications.

WWW with its standard communication protocol HTTP and user interface standard HTML [1], offers a general support for integrating applications in a common framework.

This paper describes a model of a distributed software development structure integrated in WWW. A project structure, having control on project components, is placed on a server. Project members access project components from different clients using a WWW browser. The applications that manage projects components are executed on the server site, but are invoked from clients, and their result is presented on the client site.

2. A MODEL OF A DISTRIBUTED DEVELOPMENT PROJECT

2.1. Project Structure and Software Configuration Management

A typical implementation of a software development project is a repository where project components such as documentation, data, source code files, programs, etc. are saved. Different types of repositories can be used - a relational or object-oriented data base, or simply a file system. In addition to components, a project should include information about access rights, who is allowed to modify the project components and who is allowed to manipulate with the whole project.

Project model presented in this paper uses a tree directory structure, where the top of the structure is defined as the project root and each node contain files that make a functional part of the project

A Software Configuration Management (CM) support is very important in the development process. It makes possible to keep track of different versions of software components, prevents double updating of the same version of components, groups together (i.e. configure) selected component versions into a configuration set, etc. Different CM tools are available on the market, from very simple tools managing file versions to very expensive and sophisticated packages supporting a complete development process. One of the most popular CM tools is the Revision Control System (RCS) [2].

RCS gives a basic support for a version management. Versioned files, i.e. files which contain several versions of ordinary files, are saved in RCS directories. In order to use a specific version of a file, a user must *check out* it. If a version is also supposed to be changed, it must be *locked*. A locked version cannot be locked once again, so the double updating of the same version is not allowed. When the modification of a checked-out version is completed, the user will *check in* the modified file and a new version will be created in the corresponding versioned file.

The RCS package includes the following basic commands:

- `co` - check out a file version
- `ci` - check in a file
- `rlog` - display information about versioned files
- `rcsdiff` - show differences between two versions of a file
- `rcs` - a general purpose command for versioned files manipulations

Using the RCS package a simple project model can be designed:

A project is a tree directory structure where each node contains a RCS sub-directory. RCS directories include versioned files. Nodes themselves contain specific file versions checked out from the RCS sub-directories. Typically the latest versions will be checked out. These files are used as references, but it is not allowed to modify them. However, the project members are allowed to check out files in their private work structures, modify the files and then check in them back into the project.

2.2. Using WWW and Netscape client for distributing project components

The model described in Chapter 2.1 works fine on a local system, but it is not sufficient for a work on different, geographically distributed platforms. For a full functionality a client/server support is required, which means that the project members can use, check out and check in the project components independently on which place or platform they work. Different CM tools have this support, like ClearCase [3] that contains a client and a server part of the application. However such a

support is usually limited within the tool and specific platforms. On the contrary, WWW with the Common Gate Interface (CGI) [4] scripts gives, although a simple, a general client/server support.

Using WWW features a distributed project model can be constructed: A server contains the project structure including all files in the local file system. Project members see the project structure via Netscape. Any change made on the server site is directly seen on the client site (Figure 1).

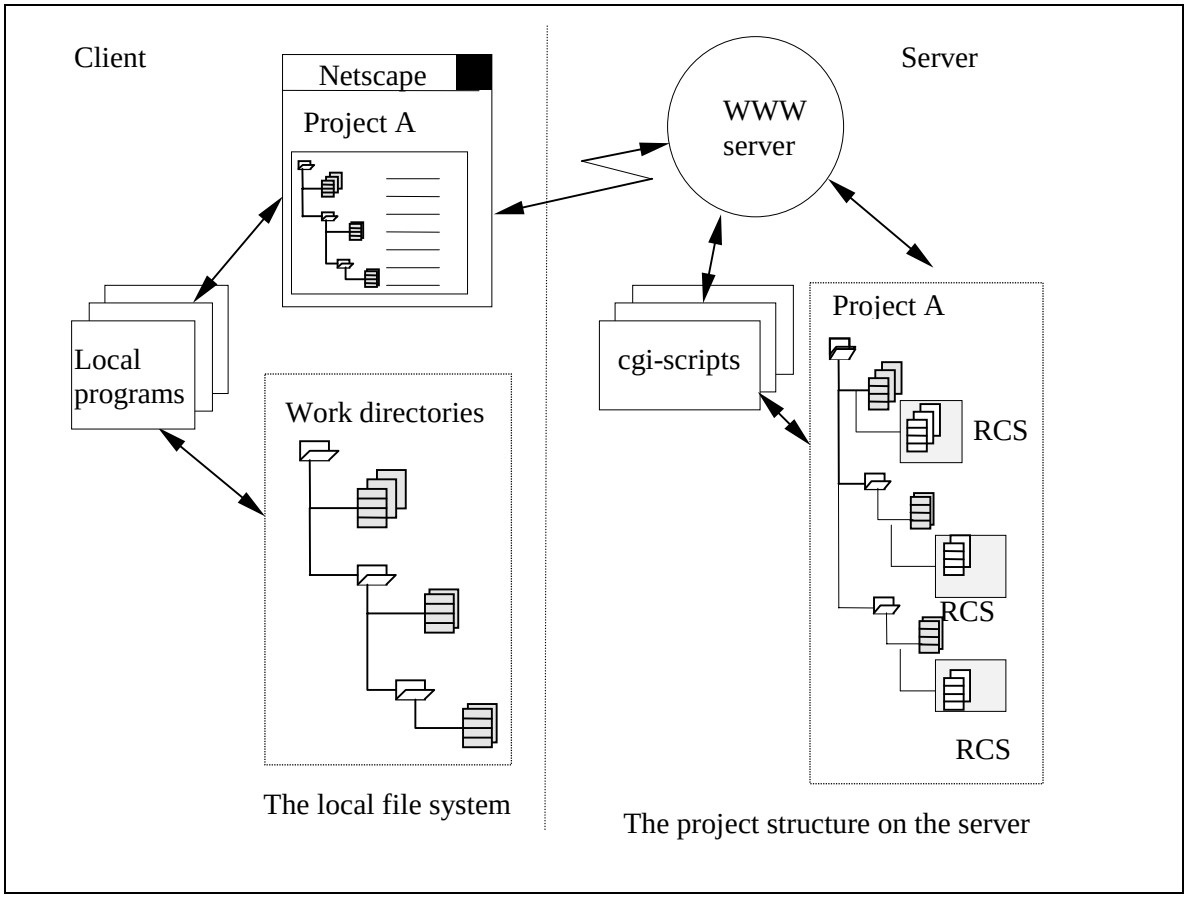

Figure 1. Displaying a project structure on a client using Netscape

Netscape or another WEB browser is used for a connection between client and server and cgi scripts executed on the server side provide services to the client.

The communication between a project member on the client site and the server happens trough the HTML pages, some of them edited and prepared in advance, but mostly dynamically created by cgi-scripts.

The following basic parts are included in the model:

1. Log into a project
2. Accessing files
3. Accessing specific version of files saved in RCS library

4. Checking out and locking a file
5. Checking in a modified file to the project

2.3 The Project Login Page

The login page is generated by a cgi-script. This script finds all projects placed on the server, and list them in the created HTML page. The page uses FORMS mark-up tags. A user that has received the generated page can select a project, enter his/her identity and password and submit the inquiry to the server. An example of the generated page is shows on Figure 2.

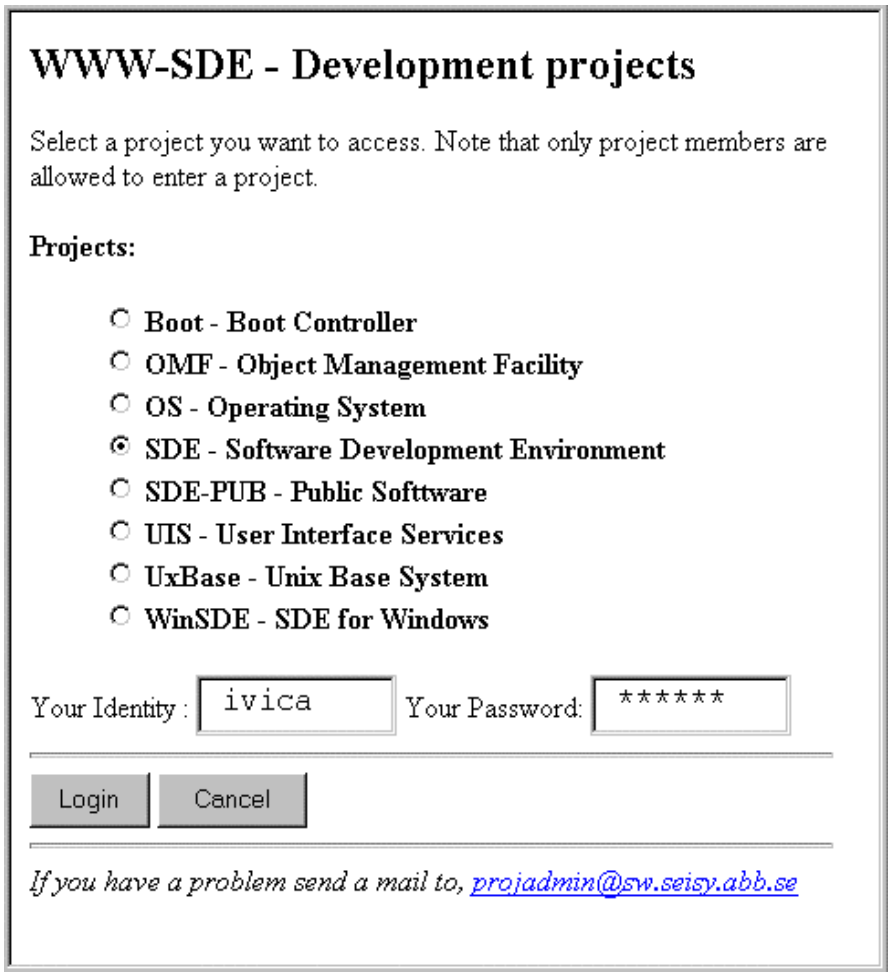

Figure 2. Generated HTML page for entering a project

The password mechanism fulfils the security basic level. Additional security may be achieved by setting different levels of limitations on the server site (like limitation on access from the Internet, access to a specific structure, etc.).

2.4 Project View

When a user sends a request for logging into a project, another cgi-script checks the user's identity and password. If the entered data fulfil the requirements, the script creates a new HTML page, a Project Page.

The Project Page includes several functions:

1. It contains links to some common project structures like project overview, messages and links to the project members.
2. It lists the project directory structure, starting from the project top.
3. For the selected directory, it shows all the files placed in the belonging RCS library.
4. The page includes a command part where a user can specify a file that will be checked out from the project structure on the server, or checked in from the client to the project.

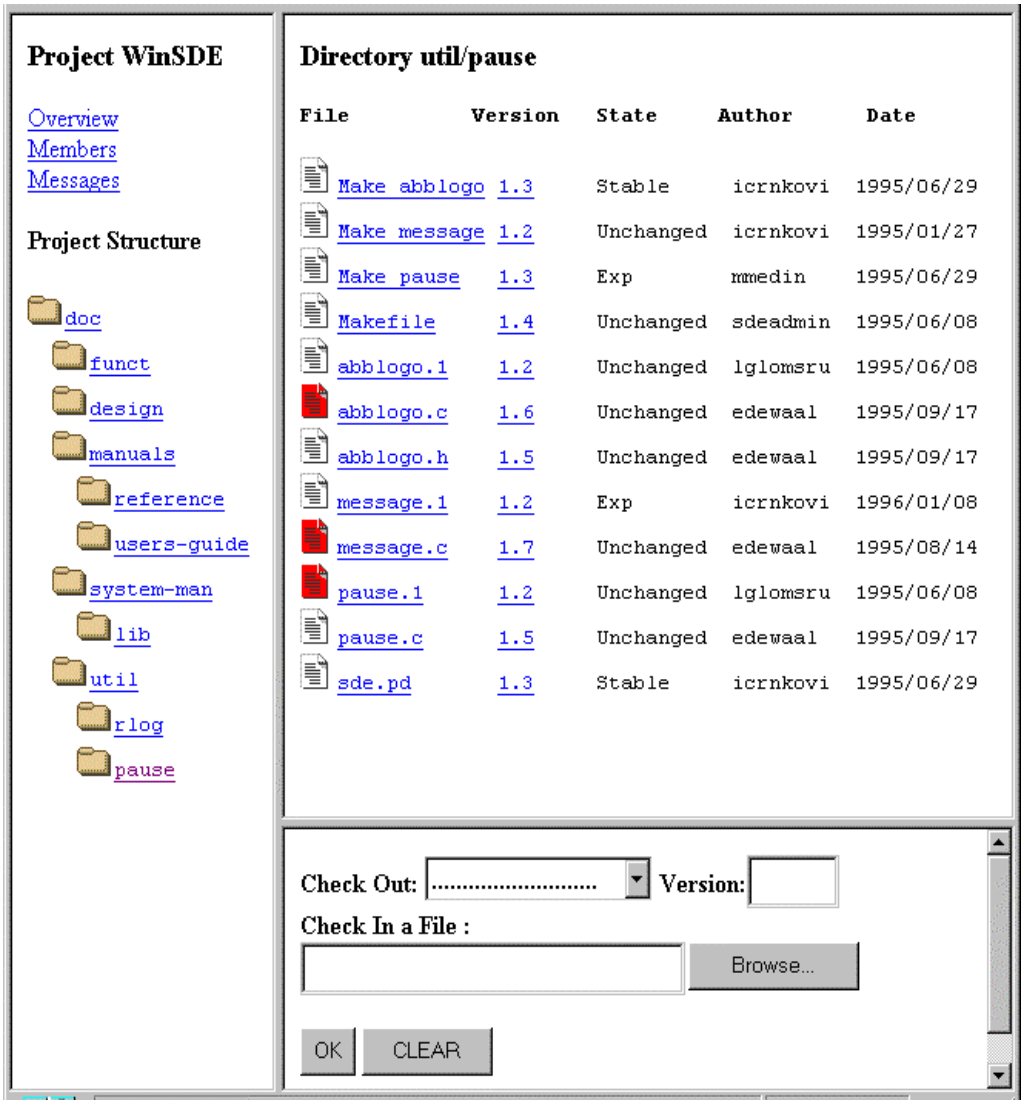

Figure 3. The Project Main Page.

As shown on the figure, some new features of Netscape 2.0 have been used. Some of these features are proposed as extensions to HTML 2.0 [5]: For the organisation of the complete page, the FRAMES feature is used. Three frames, the *project-dir*, *project-files* and *commands* frames are created. Each frame is generated by a separate cgi-script.

The project-dir frame shows the project directory structure. Each directory is implemented as a link to a cgi-script that generates a list of files in the project-files frame.

The second frame lists all files present in the selected directory. Different colors of the file icon show if a file is locked or not. In addition to the file name, the current file version, its state and its author are displayed. Both file names and file versions are implemented as links. When clicking on a file name, the selected file is checked out from the project on the server and displayed on the client site. A Netscape standard mechanism for launching associated applications is used. The version link is assigned to another cgi-script that build a history page: all versions saved in the RCS sub-directory of the selected file are displayed. The format of the history page is the same as the project-files page. Once again, it is possible to click on any version by clicking on it.

The third frame includes a FORM type of page. It enables a user to specify a file version to check out and lock it, or to check in a file. The proposed extension to HTML 3.0 [6], HTTP File Upload feature is used. The forms include a new attribute ENCTYPE which makes it possible to browse on the local file system, select a file and send it to the server.

The command page in the third frame is also dynamically built, because it includes a list of all files present in the selected directory, and this list is changed when selecting different directories.

2.5 Limitations in Graphical User Interface

When designing a highly interactive applications based on HTML standard we meet some limitations. HTML is designed for an access and a presentation of information, not for the interactive exchange of information. For example, a “classic” graphical application has a possibility to first select an object and then apply different activities on it. The object selection is not necessary connected to an immediate action as is the case when using HTML. In the present model this problem is circumvent by creation of several links to the same object but using different actions. One example of this case is when a file name link invokes a cgi-script that check out a file version, and the file version link invokes another cgi-scripts that shows the file history.

However, these limitations disappear with integration of Java [7] and OLE Automation in Netscape.

3. CONCLUSION

The model presented in this paper is an example that shows the possibilities of lifting a local application to a distributed one with a simple, but effective integration in WWW. It is not necessary to change applications themselves, but simply encapsulate it in a WWW framework.

There are several aspects of integration in WWW: A development of a graphical user interface is independent on platforms, applications get a common user interface, they become available on different platforms, distributed data can be accessed using standard protocols, and, probably the most important, very different applications can be combined together. In the nearest future we can expect a wide use of WWW not only in different fields like multimedia or software development, but also their integration.

4. REFERENCES

[1] Hypertext Transfer Protocol, *http://www.w3.org/hypertext/WWW/Protocols*, 1996

HyperText Markup Language, *http://www.w3.org/hypertext/WWW/MarkUp/*, 1996

[2] W. F. Tichy, "RCS: A System for Version Control", *Software Practice & Experience*, vol.15, no 7, July 1985

[3] SCM Specification / Evaluation Guide, *http://www.atria.com/Literature/w_p/htm/scmseg.html*, 1996

[4] The Common Gateway Interface, *http://hoohoo.ncsa.uiuc.edu/cgi/overview.html*, 1996

[5] Extensions to HTML 2.0, *http://home.netscape.com/assist/net_sites/html_extensions.html*, 1996

[6] Extensions to HTML 3.0, *http://home.netscape.com/assist/net_sites/html_extensions_3.html*, 1996

[7] James Gosling & Henry McGilton, "The Java(tm) Language Environment: A White Paper", *http://java.sun.com/whitePaper/java-whitepaper-1.html*, 1996