Building an Web-Based Open Source Tool to Enhance Project Management, Monitoring and Collaboration in Scientific Projects

Alexander Schatten[1], A Min Tjoa[1],
Amin Andjomshoa[2], M. Hassan Shafazand[2]
[1]Institute of Software Technology and Interactive Systems
Technical University of Vienna
{schatten,tjoa}@ifs.tuwien.ac.at
[2]University of Kerman, Kerman, Iran

Abstract

Scientific Cooperation, especially when co-workers and teams are dislocated, requires support for project management to distribute information to all members of the team. Such a support should allow project monitoring for team leaders and the management of resources, communication and documentation.

In early 2000 the Austrian Federal Ministry of Education, Research and Culture launched a project for the management and support work of bilateral scientific projects between Austria and partner countries. This paper will describe the architecture and results of this project.

The Institute for Software Technology develops an open source web application software; to enhance these co-operative tasks. The „Scientific Workplace"-tool allows management of users and skills, projects, tasks, to-do lists, resources (e.g. documents) and access control, project monitoring and messaging.

The software will be distributed as open source application and is built itself upon open source server tools (application server, database, web-server) and uses open communication protocols (SOAP) and XML file-formats. The server application as well as the administration application and the user front-end are implemented in a platform independent way. This way of realisation increases the flexibility of use as well as the spread of the application as there are no hurdles in terms of license fees or option to make modifications/adaptations.

1. The SWP Project

The Scientific Workplace Project (SWP) was initiated by the Austrian Federal Ministry of Education, Science and Culture to perform a web-based management tool which is suitable for scientific collaboration in projects which are performed in different locations. Especially SWP should be used for joint projects with scientific teams of other countries. The SWP project is performed jointly between the Institute of Software Technology and Interactive Systems at the Austrian Side and the Computer Science Department of the Kerman University at the Iranian Side, which is represented by TCO (Technology Co-operation Office of the Iranian Presidency). The project starts in early 2000 and a very promising prototype has been finished by July 2001. The aim of this paper is to describe the very successful joint work between the Iranian software engineers

headed by Prof. Shafazand from Kerman University and the Austrian team. Within the Austrian team most of the design and implementation work has been performed by Alexander Schatten. In Iran most of the design and implementation work has been performed by Amin Andjomshoa. All the design and platform decisions have been taken jointly by both teams. We plan to continue the project for another two years with the goal of having a public-tool available for joint distributed scientific work with features yet not commercially available.

2. Collaboration

2.1. Dislocated Workgroups

Projects with workers in dislocated workgroups are a specific target for SWP. Already in "conventional" projects, communication and exchange of information is a crucial factor for success; In the case of dislocated workgroups the complexity of this problem is obviously increasing.

Many of the collaborative tasks can be co-ordinated using email and telephone, but if more than two or three workers are involved, a more organised and centralized information store shows to be useful. The SWP platform helps the project members in organizing project members (user management, information), communication (discussion groups, news system, chat, messaging support) and project management (project data, tasks, to-do lists, resource management).

2.2. Project Management

2.2.1. Introduction

The support for project management and monitoring is an essential part of the SWP system. As shown in Fig. 2 and explained in the Example section, projects can be defined in the SWP context. First of all, each project can contain „general“ project information like project name, start date, end date, description and the like.

More important is the feature to build a „work tree“ by defining tasks and subtasks for each project. These tasks are the centre of the project management. Each task has assigned a user who is responsible for this task. He also defines the progress settings of his work.

2.2.2. Project Monitoring

For project monitoring, the system offers visualisation functionality that displays the progress of a complete project or of several main tasks be combining the progress information, start, end date and other attributes of the sub tasks. This helps the project manager to keep track of the work of his/her specific project and the „top-manager“ to keep track of all projects running in his/her company or institute.

2.2.3. Support for Collaboration

Several collaboration features can be assigned to each task: Users can define and manage „to-do“ lists and keep an overview over the work of the task. Arbitrary resources can be uploaded to the system and assigned to specific tasks. This helps to organise project documentation in a natural way (as assigning them to the tasks they belong too).

All members of a project have read access to the resources of a project, hence this feature enhances collaboration as besides the project management functionality an organised resource library is provided.

2.3. User Management

Of course a user management module is implemented in the SWP system. Administrators may add new user accounts; project manager can add those users to their projects as project members.

Besides the account information, specific user data is acquired like contact information, skills, etc.This helps (as described below) to increase the communication facilities when co-working on projects.

2.4. Communication Facilities

2.4.1. Introduction

Communication is another important factor in collaboration, especially when the teams are dislocated. We support three types of communication at the current release:

2.4.2. Discussion / News

A discussion/news mechanism is implemented, that allows to discuss certain topics considering a specific project or the complete SWP working context (e.g. the complete institute) or to allow project managers to post regular „newsletters“ to the project team about essential developments in the project. These messages are stored on the server. A user can post messages to all SWP user or to a specific project.

2.4.3. Messaging

At the current version we tend to think that users prefer to use the email clients they are used to. Hence we do not implement some kind of email client, though this is an option for future releases.

Nevertheless: In the SWP system data of all users are available for all SWP members, naturally we support sending of emails via web-front-end. A user may send group mails to all SWP members, to specific users and to members of a project or to a combination of them. This leverages sending „group“ messages.

2.4.4. Chat

A simple chat module is implemented in the first version, which could enhance specific work and discussions in some projects.

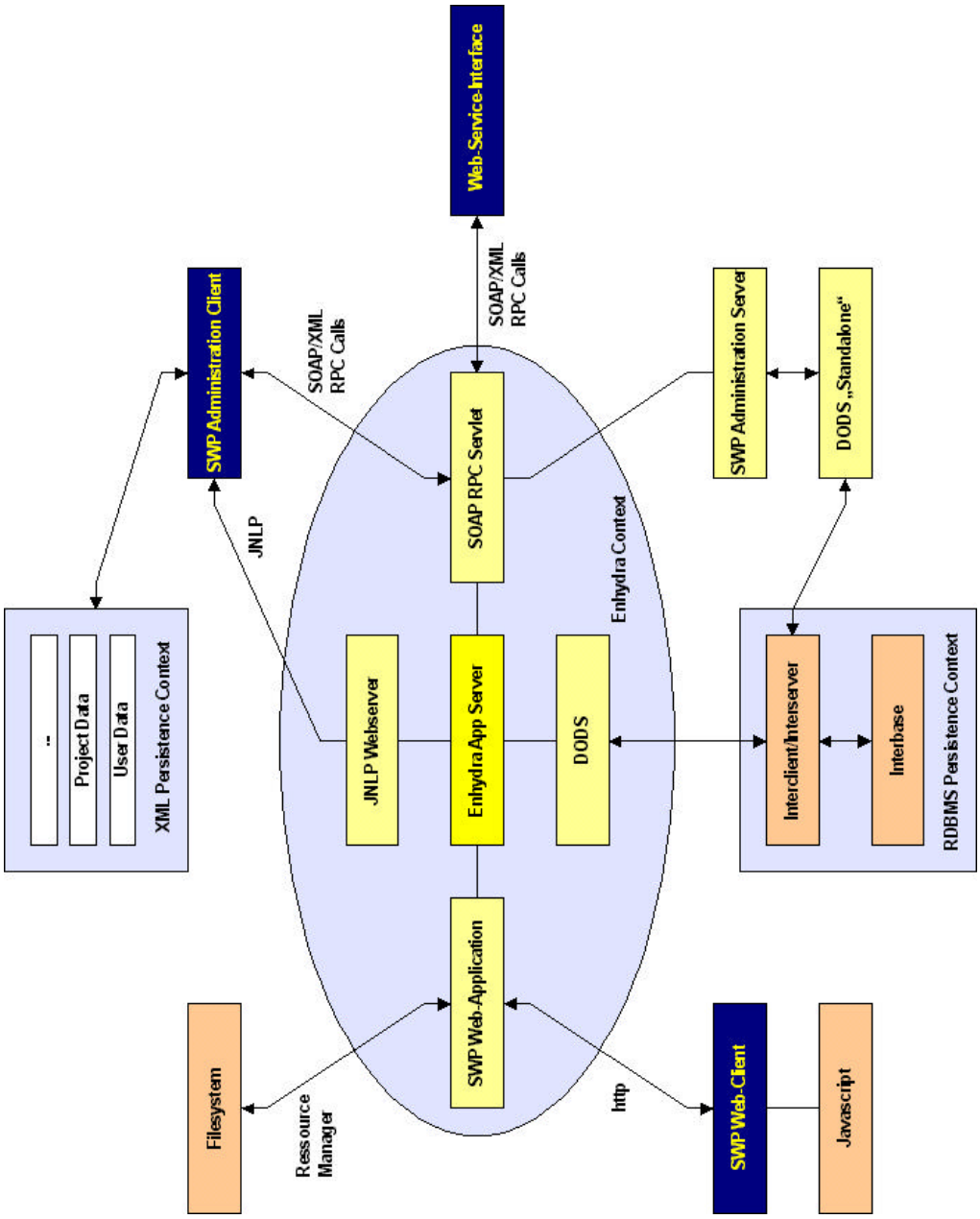

Figure 1 Technical Overview about the SWP Infrastructure

3. Technical Overview

3.1. Overview

An overview of the SWP infrastructure is given in *Fig. 1*.

3.2. Persistence

3.2.1. RDBMS

The data of the SWP system is stored in a RDBM system with the exception of the resources (files) uploaded to the system. As the access to the database system is done using an object relational mapping tool, the underlying database system is not limited to a specific system. Currently we use the open source version of Interbase [7, 8], as a high performance RDBMS with all necessary functionality running on Linux as well as on Windows and Solaris.

3.2.2. DODS

Data Object Design Studio (DODS) is part of the Enhydra application [7] server and is an object-relational mapping tool. Using DODS allows to design the persistence (database) layer in a database neutral way and simplifies (Java) access to the RDBMS. DODS can generate database metadata for different database systems like Interbase, PostgreSQL, Oracle and so on. The combination of DODS and the XML mapping system allows easily to adapt a running system to another database system by making a backup to XML, changing the database and restoring the XML data to the new database system.

3.2.3. XML

Mapping tools exists that allow import and export of the complete SWP data (stored in RDBMS) to XML [3, 4]. This has the advantage, which this kind of persistence is platform and database neutral and can be used to backup and restore data or exchange of data between different (SWP) systems, for example using the SOAP webservice interface.

3.2.4. Filesystem

All "core" information are stored in the RDBM system as mentioned above. The SWP system allows also storage of resources (file upload) to assign documentation and other stuff to certain tasks. These files are stored in the server file-system, not in the database management system.
Only references are stored in the RDBMS. At the first glance it could seem more "natural" and consistent to store this binary data in the RDBMS too. We decided to store this kind of data in the file system, as it could have serious impact in performance when using the RDBMS for storage. Moreover this data is used "as is" and the database system offers no additional functionality, which means, that the access to these files would become more difficult than necessary. Furthermore this approach seems to be more flexible as for storage of binary data (files) multiple hard-disc can be defined and the database files stay more compact and easier to handle.

3.3. Web-Application

The "normal" interface of the SWP system is a web-browser using HTML and Javascript functionality. The Enhydra open source application server [7] is used to generate this user interface for interaction with the system. This approach allows flexible access to the SWP system as there is no need to install specific client applications. A SWP user can access his/her data from all client computers that have an internet connection and a web-browser. The interface is functional but light-weight and therefore offers the functionality also to clients which have a low-bandwidth access e.g. using modems.

3.4. Administration

3.4.1. Application Server

The administration of the application server is done using the Enhydra application server interface. This is "low-level" administration and is usually only necessary when installing the system or shutting down (parts) of the system.

In the Enhydra context the web-application, the SOAP service [11] and the JNLP [9, 10] webserver to provide the SWP administration interface is running. These systems can be configured individually and can also be stopped selectively. For example: for "high security application" the administration access could be stopped at a "low level" by disabling the SOAP service in the application server.

But the system is again more flexible: Enyhdra is needed to serve the Web-Interface for SWP. All other services could be installed in arbitrary other Webserver/Servlet containers, even on different server systems. This could increase the security and performance of a complete SWP system.

3.4.2. Database Server

The SWP data is administrated using the SWP-Administration application. Obviously all RDBM systems have their own administration applications to make low level modifications like tuning set-up. Usually this is only required when creating the database on installing the SWP system and eventually to make performance tuning steps.
Also the backup mechanisms of the RBDMS systems can be used (consider a Oracle or DB2 installation in a company or institute, that serves different databases and one of these databases is the SWP database. Then a backup mechanism may be installed already, and using this mechanism may seem natural.)

3.4.3. Web-Application-Administration

A high-level administration tool to administrate the SWP functionality is available as Java (JNLP) application [9, 10]. We decided to write a Java application - as a richer interface for administration is desired - and administrative tasks will not have to be necessarily available "everywhere". This application is served by Java Webstart that is a mechanism that automatically distributes the application to the admin clients and guarantees that always the most recent version of the application and the libraries are used for administrative functions.

3.4.4. Web-Service Interface

We implement the open standard SOAP (supported by all major software companies) [11], which is a network protocol based on XML that supports remote procedure calls via http.

The consequence is, that we offer basic functionality to exchange data with the SWP system using the Apache SOAP implementation. This can be used to „plug-in“ several client applications speaking the SOAP protocol to exchange data.

The admin client connects and communicates with the server system using the open SOAP protocol (see Webservice Interface) and can add additional functionality like other clients for users, backup mechanism, batch jobs, and so on.

Currently we use the SOAP interface to connect the administration client to the SWP server.

3.4.5. Logging

A logging mechanism is implemented to store certain user actions like logging in, logging out, trying to log in with wrong passwords, starting backup and so on. The logging data is stored in the database and can be queried by the administrator.

4. Public Services

4.1. Introduction

Parts of the information stored in the SWP system can be used for a "public view": Usually companies or academic institutes provide contact and project information on the website. This can be done automatically using the SWP system.

Part of the user information and attributes like names, description and members of the current projects can be published as a public view. This helps keeping these information up to date as a modification of email addresses, telephone numbers or workgroup members are "synchronised" with the internal SWP system.

The Project View shows the data of all SWP projects, but only at a high level: e.g. project name, start/end, user assigned to project...

The person overview shows all active persons of the SWP context with contact information and projects the user is assigned to.

If a logged in user accesses these information pages, he has access also to „internal“ information like telephone numbers, address information, etc.

5. Example

In the example in *Fig. 2*, an SWP server instance is shown. The example is kept simple enough to prevent loosing the overview, but nevertheless it is complex enough to show the SWP access functionality.

This *server* could be located in Austria.

Three projects and eight users (4 in Iran, 4 in Austria) are defined in the SWP context. Two users (*Iran-User 1 and Austria-User 2*) are declared as system administrators. Hence they can administrate the system: add/remove user, add/remove projects, control the backup system and so on. These administrative tasks are performed using the administration application via SOAP

interface. (Being administrator does not mean necessarily to have access to all projects using the "normal" web-interface.)

Project 1 has two main tasks, Task 1 consists moreover of two subtasks. As a next step we will take a closer look at the user / project relations:

Iran-User 1 (IU1) is manager of Project 1: This has the consequence, that he is responsible for the project. A project has exactly one manager (or none, as Project 3, which is declared, but obviously has not started yet). IU1 obviously has access to all project 1 data (including tasks and subtasks), but has in this example no access to data of the other two projects.

IU2 is manager of Task 1, hence responsible for Task 1 and subtasks 1 and 2 (!). He can declare the task as being finished, but this has to be confirmed by the project manager IU1. Furthermore he has (as user of project 1) read-access to the complete project 1 and write access to Task 1 and subtasks.

IU3 is member (worker) at subtask 2; also Austrian User 1 (AU1) is declared as worker on subtask 2. He has read access to complete project 1 and write access only to subtask 2. Each task or subtask can have more than one user assigned.

Subtask 1 has no assigned users at the moment, so it has not been started yet. Probably other users currently working on other tasks will take this tasks after having finished the other work. To finish Project 1, all subtasks and tasks have to be finished, the task managers have to declare them finished and the project manager has to confirm this.

Now let us take take a look at the Austrian user group:

AU1 (as mentioned already) is assigned to Subtask 2 of the Project 1, which is managed by an Iranian manager. He may write to subtask 2 (modify to-do list, add resources, write messages ...) and read data of project 1. AU1 is also assigned as user to project 2, but not as administrator for project 2. So he will be able to read all data of project two, but has no write access to any part unless he will be assigned to some task (no tasks are defined currently).

AU3 is declared in the SWP context, but not assigned to any project. The meaning is: AU3 could be member of the institute but is not working on any current project. He consequently has neither read nor write access to one of the three projects, but may send messages to other SWP user. As soon as a user is entered to the SWP context, he is available as worker for projects and tasks.

So *project 1 is a collaboration project* managed by an Iranian worker; work is done by Iranian and Austrian workers.

Project 3 is defined in SWP, but has neither a manager nor users assigned at the moment.

What is left is the *"top-manager*" view. Both partners have declared a "general manager", which could be the head of the institute. These managers are declared for the SWP context and have read access to all projects and additional functionality for project monitoring of many projects (GUI support to visualise progress of projects).

All mentioned access control rights concern the access to specific task/ resources. Off course all members of a project may post news for the project use the messaging or the chat system for collaboration.

All users in the SWP project work with the system using a web-interface with login function to control the user access. Furthermore there is a public view (via Internet) without login. This public view can be used to show parts of the user and project (meta)information to the "outside" world. This is useful to provide contact information and information about running projects on the institutes/companies. This feature is optional.

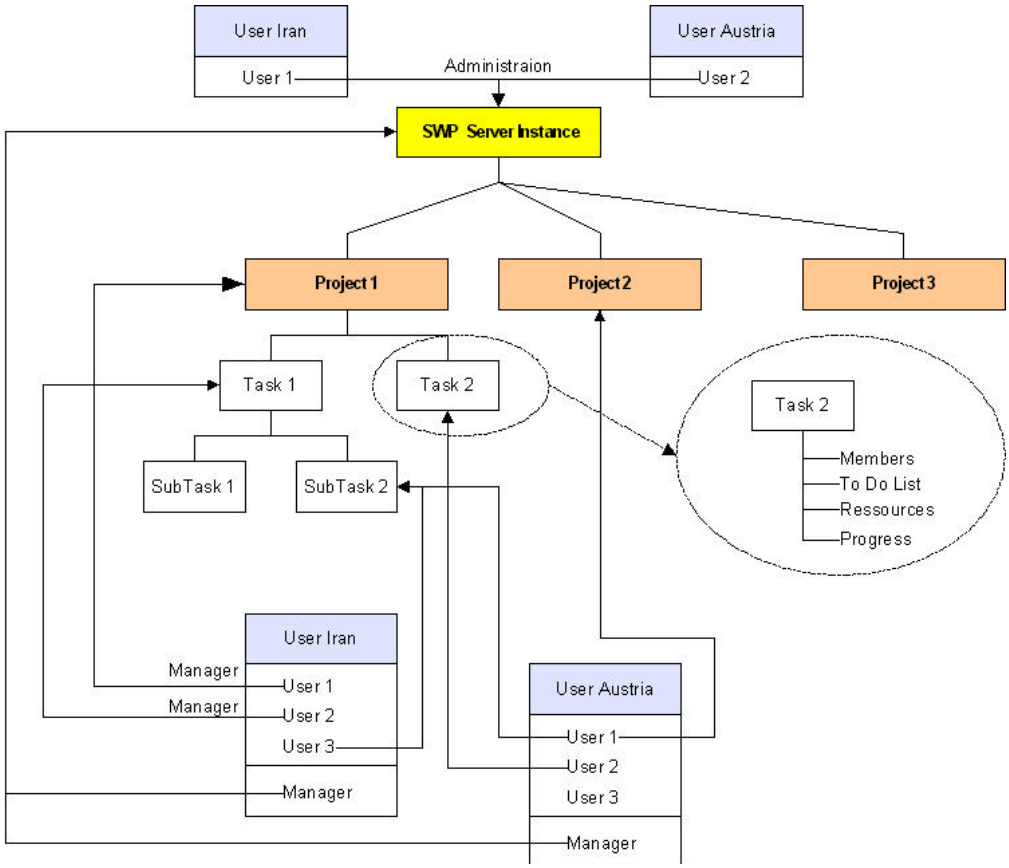

Figure 2 Example SWP Project with Users

6. Open System

6.1. Open Source

6.1.1. Introduction

After an extensive period evaluating different server systems and tools, we decided to build our system on open source tools by many reasons:

For all parts of the project professional open source systems could be found

- We believe in the impact of open source software; As SWP itself will be "distributed" as an open source system (at least the main system) it would be problematic for potential (academic) user groups to build the system on top of commercial frameworks-as the SWP system itself would be available with no license feesIf the underlying application server, database and other important components are commercial systems this would limit the distribution of our system. Especially as application servers and database systems tend to be extremely expensive.
- Building the complete system as open source tool also encourages potential users to adapt or extend this system to individual needs and eventually receive additional developments from other groups to be implemented into the "main" SWP distribution.

- Many users (in companies or in the public sector) are aware of security and transparency of such an infrastructural system. Providing the system including all sources allows these users to evaluate the crucial parts of the system and check all functionalities if they fit into their company policy or eventually modify or extend the system to fit their needs.
- Since different other infrastructures may be available (containing data that may be useful also inside the system) keeping SWP open in many directions (web-service interface) also allows us to integrate legacy systems and to easily develop modules connecting to the web-service interface to let available systems communicate with SWP server.

6.1.2. Application Server

We decided to build SWP on top of the Enhydra application server [7]. This is a Java-Open source application server supporting three-tier web applications, namely presentation layer (with xmlc), business layer (Java components) and persistence layer (object relational mapping - DODS).

6.1.3. Distribution of SWP

As the SWP system is build upon open source tools and will be distributed itself under an open source license, probably GPL [13], but this is yet under discussion. However, there will be no need for license fees for potential users, compared with commercial products.

6.1.4. Persistence/RDBMS

Using the object-relational mapping tool DODS (Enhydra app server) we are not limited to a specific database system. For our SWP distribution we currently use the Interbase RDBM system. Using other (eventually available) RDBMS like Oracle or DB2 is possible.

6.1.5. Persistence/XML

Parallel to the storage in an RDBMS all data can be mapped to XML. This is useful as it allows a platform and system independent backup/restore or archive mechanism and leverages exchange mechanism between systems (multiple SWP systems or exchange between an SWP and arbitrary other systems).

6.1.6. Transparency

This open source strategy can be important for user as it guarantees

- Extensibility: Users can extend the system where they see the need to.
- Adaptability: User can adapt the system to existing infrastructure (databases, ...)
- Security: Security aware user can examine the SWP sources and analyse all functions that could be possible security hazards.

6.2. Open Protocol

It is an important task to keep SWP open in many ways; not only considering open source, but also communication facilities. We decided to provide a SOAP web-service interface using Apache SOAP [11]. Through this interface all relevant data can be retrieved or stored in the SWP system. Currently we also provide one client application used for system administration.

Basically arbitrary other clients can be written that connect to this open interface. E.g. applications to integrate SWP in existing infrastructure (import/export data) or write alternative client applications to access the SWP data and functionality.

6.2.1. Web Service Architecture

The Web-service architecture was already explained in detail in the technical section, however it is used, or can be used in future applications:

- Administration Interface uses SOAP service
- Extensions can be written by other users (batch mechanism, integration of other systems)
- Synchronisation of different SWP server could be done

6.2.2. Xhtml – Webapp

The data produced by XMLC (Enhydra) that generates the web-user interface is xhtml.

6.2.3. Backup/Restore

Backup and Restore can be done using the usual database backup mechanism and additionally in a platform independent way using the XML import and export functionality of SWP.

6.2.4. Modelling / Documentation

Also the documentation is provided in standard formats: UML (Use cases) and RUP (Rational Unified Process) [15, 16] is adapted to provide high-level documentation for users who want to extend or adapt the system or add additional functionality.

Currently the use of O'Reilly Docbook [14] and XML/XSL [5, 6] is evaluated to be used for project documentation and probably also for automatically generated reports done by SWP system.

7. References

[1] Bray, Tim and Sperberg-McQueen, C.M. (1996), Initial XML draft, http://www.w3.org/TR/WD-xml-961114.html

[2] Harold, Elliotte Rusty (1999), XML Bible, IDG Books

[3] Bob DuCharme (1998), XML: The Annotated Specification, Prentice Hall PTR

[4] Sharon Adler, Extensible Stylesheet Language Xsl : Version 1.0 - W3C Working Draft 27 March 2000, iUniverse.com

[5] James Clark, XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 November 1999, http://www.w3.org/TR/xslt

[6] Enhydra Application Server: http://www.enhydra.org

[7] Timothy Dyck (2000), ZDNet eWeek, Interbase proves it’s mettle, http://www.zdnet.com/eweek/stories/general/0,11011,2436153,00.html

[8] Interbase: http://www.borland.com/interbase, http://firebird.sourceforge.net

[9] Mauro Marinilli, Deploying Java Applications using JNLP and Web Start, Sams (in Print)

[10] JNLP Sun: http://java.sun.com/products/javawebstart/

[11] Apache SOAP: http://xml.apache.org/soap/index.html

[12] Wrox Multi Team et.al. (2000), Professional Java Server Programming J2EE Edition, Wrox Press Inc

[13] GPL, http://www.gnu.org/copyleft/gpl.html

[14] Norman Walsh & Leonard Muellner (1999), DocBook: The Definitive Guide, 1st Edition, O'Reilly

[15] Geri Schneider, Jason P. Winters (1998), Applying Use Cases, Addison-Wesley

[16] James Rumbaugh, Ivar Jacobson, Grady Booch (1998), The Unified Modelling Language Reference Manual, Addison-Wesley