

Edukalibre system architecture

Diego Chaparro

Luis Cañas

Revision History

Revision 0.2 10th February, 2004

Note: This document belongs to Edukalibre project (<http://edukalibre.dat.escet.urjc.es>), financed by the European Commission (MINERVA).

1. Scope of this document

This document will show the requisites, design and architecture of the edukalibre system, which will allow to create educational material in a collaborative way between teachers, students and other possible contributors.

This document will define the existent actors in the system, the architecture, the design and some possible scenarios. But this is only a proposal that will have to be discussed to arrange a consensus, and to decide about some parts of the system.

2. Requisites/Functionality

Requisites/Functionality of the system.

- The system will be able to lodge documents and to schedule their control version.
- The system will authenticate the users, assigning two different access. The first user group could be called group of publishers, and will have writing permission on its published document. The second user group (contributors) could be formed mainly by readers, but potentials publishers too; the main difference between these and the publishers of the first group is that its modification in documents must be allowed by the first group.
- The system will offer the document in different formats; it will make edition much easier..
- The assistant (a publisher) will be able to publish the document in several ways. By means of OpenOffice (for instance), the assistant will upload his document to Edukalibres system in a way which makes its use transparent for the user.
- The publisher will be able to get it through the Web site, being able to download the version that it wants. In the same way, the publisher will be able to see via Web the different versions of a document.

- The system must give the control on the modifications of a document to its publisher. For instance, if a user of readers group try to upload a modified document, the system must announce it to the publisher, so that this one decides on its possible update.
- The system will make possible the publisher to obtain the document in several formats.. The assistant will obtain the document in doocbook format or directly interpreted by OpenOffice.
- The contributor will have access to the system and will be able to download the document that it prefers. The contributor will be able to consult the differences between the last offered versions.
- The contributor will be able to change its local copy of the document and to save the different versions made. Once the document has been modified, the user will have the chance of upload this to the system; but this last one must be allowed by the creator/publisher.

3. Design/Architecture

Design/architecture of the system.

3.1. Actors

Sytem actors:

Coordinator

Is someone that can coordinate the edition of a document, it decides who is inside the group of authors and decide when is released a new version of a document. Each document has a coordinator or a group of coordinators.

Author

Is the person who create a document in the system. He make different versions of the document, and has the choice of accept or deny the contributions of other people over his document. It can be a single author or a group of authors. People can be included in the group of authors.

Contributor

Is the person who uses the system as a reader, and has the chance of modify and create his own versions of the document. These modifications can be included in the main branch of the document, and contributor can be included in the group of authors.

Reader

Is the person who uses the system only as a reader. There are two kind of readers:

- Online reader: it accesses to documents in an available format to be read on-line, and can access directly to modify the document as a contributor.
- Offline reader: it access to documents in a format available to read off-line, such as pdf or in paper.

3.2. Architecture

The system have many components, in the server side and in the client side:

- Control Version System (Subversion or GNUArch)
- Access scripts module: a group of scripts to access to the repository over ssh: checkout documents, commit documents, etc.
- OpenOffice module: a plugin to OpenOffice to edit the documents and publish these in the system.
- WebDav interface to access the repository through a WebDav client.
- Compilation module: a group of scripts to compile the documents in the repository to pdf and html formats.
- Objects in the repository: there can be some kind of objects. DocBook/XML or Latex at first, but then we can add some other objects, like Octave.

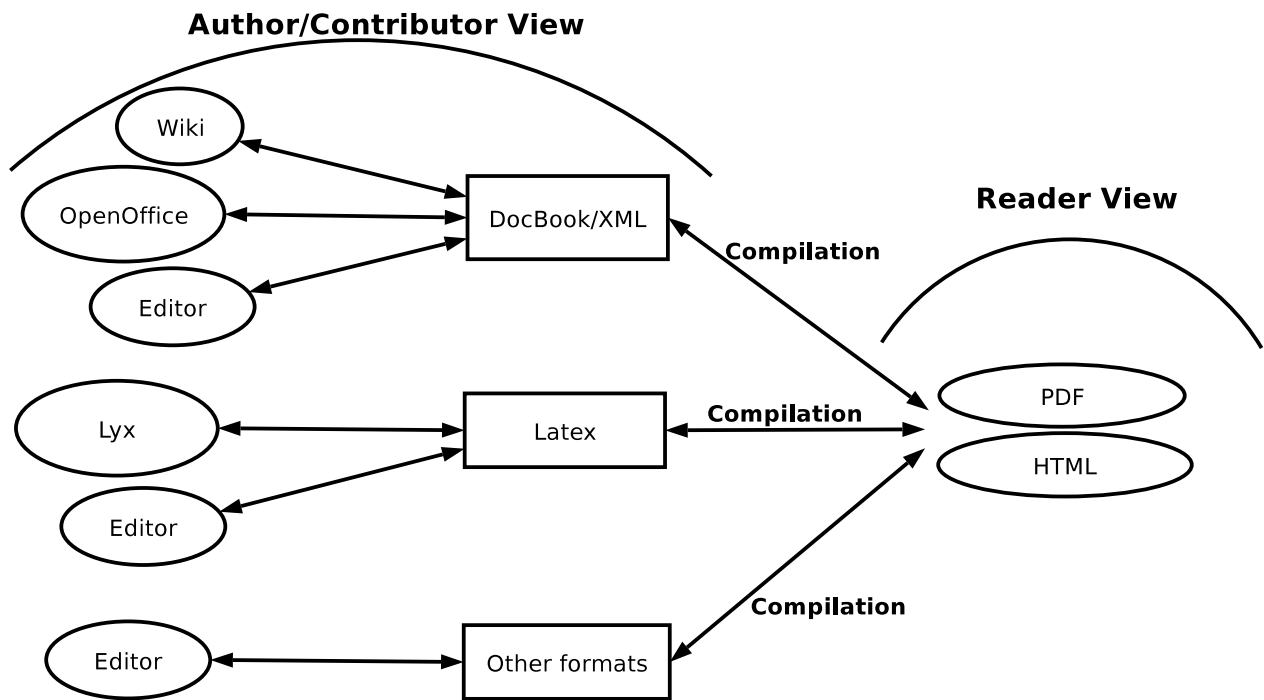
3.2.1. Architecture of formats

The formats are very important in the system, because there are a few formats in which a document will be converted dependent of the action made over the document. There are a few formats that will be used during the document edition, some other formats in the document read, etc.

We also need some tools to make the conversions between documents. The ideal situation is that all the formats can be converted from the base formats in the system, that will be DocBook and Latex at the beginning, but maybe some other will be added. The rest of documents will be converted from the base formats.

In the next figure we can see the internal architecture of the formats in the system.

Figure 1. Architecture of formats



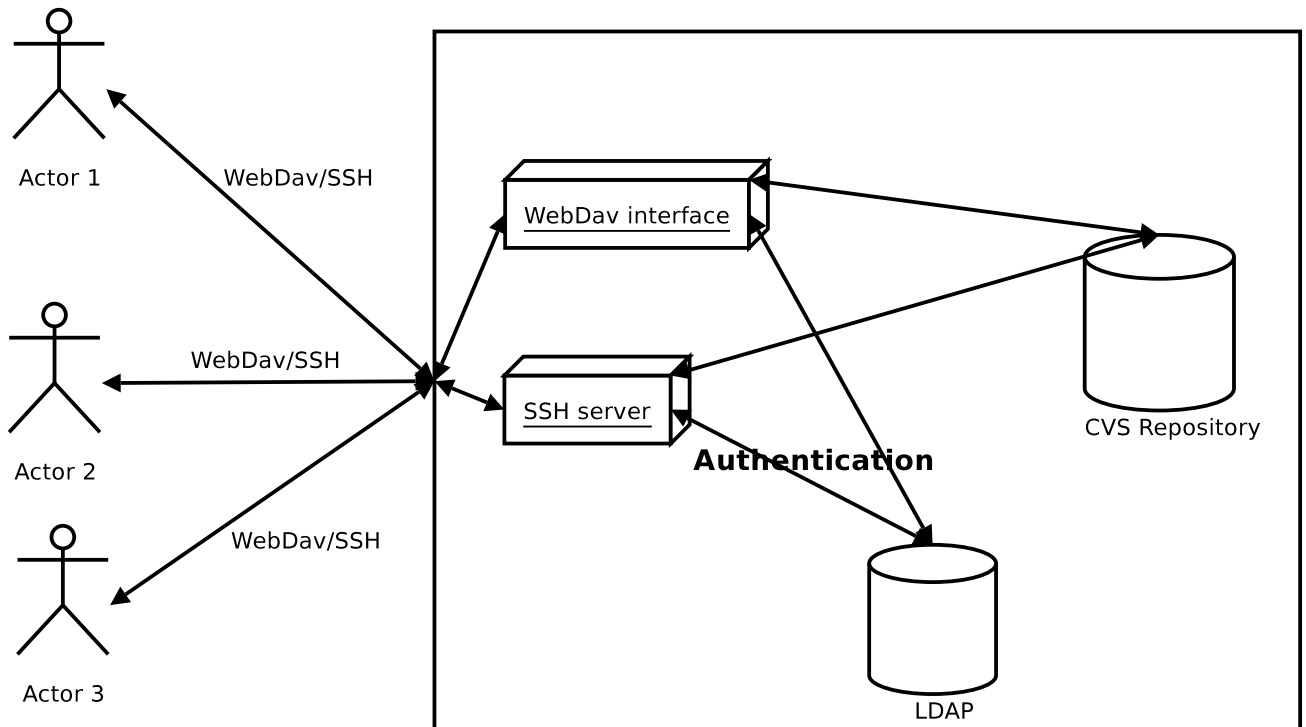
3.2.2. Architecture of applications

There are a set of applications that will be used in the system:

- LDAP for system authentication
- CVS to store the documents
- SSH and WebDav to access the repository
- Scripts to convert documents between formats

In the next figure we can see the internal architecture of the applications in the system.

Figure 2. Architecture of applications



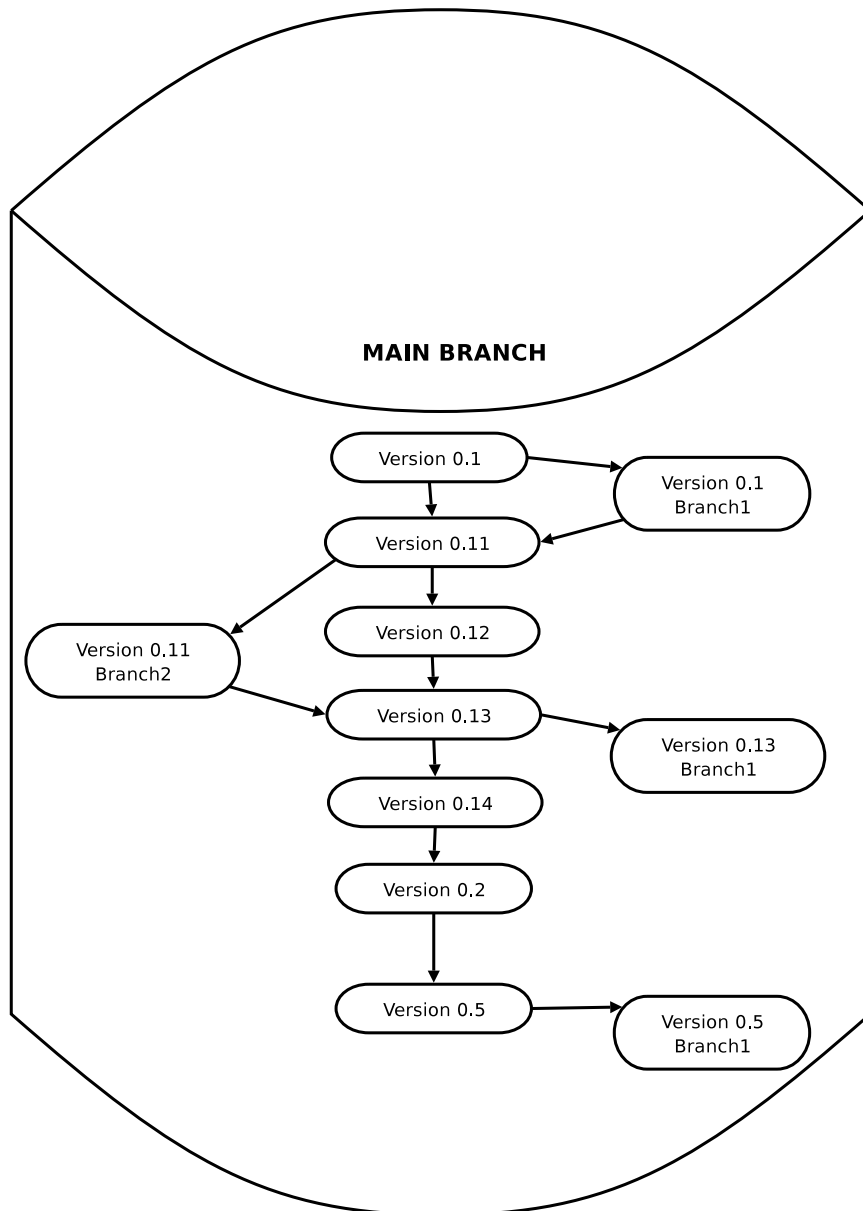
3.2.3. Architecture of control version system

The control version system is a very important part in the system, because all the functionality works around how the documents are stored, how new branches are created, how some branches are joined in only one, and many other actions over the documents.

A document will have a main branch, which is created and updated by the authors. Then when a contributor update the document a new branch is created and the authors will decide if the new branch will be included in the main branch or not.

In the next figure we can see an example of a document creation, with his main branch and other branches that are created by the contributors, some of them are included in the main branch.

Figure 3. Architecture of control version system



4. Example scenarios

We will describe here some of the possible scenarios which we will find when the system will be ready.

Usually, these scenarios have:

- One or more authors

- Educational material
- One or more contributors
- One or more final and partial versions of the document

- *Scenario 1*

A teacher in the University X in XX is preparing a document about “Advanced Algorithms” for a postgraduate course. He is writing his document in DocBook using OpenOffice. He publishes the version 0.1 and 0.2 in the system.

Then a teacher in the University Y in YY sees the document on the project web page and decides to use the document for a course that he is teaching. He makes some changes in the document and sends these changes to the system. The original author X sees the proposed changes and integrates them into the main version of the document. After some modifications done by the original author and some contributions by the other teacher, the versions 0.3 and 0.4 are published in the system. Then the contributor is converted into an author because he wants to participate actively in the development of the document.

Later, two teachers in the University Z see the document and they decide to collaborate with it, but the teaching language in their University is different from the original. The system allows to generate parallel versions of the same document in different languages, and they decide to maintain the document version in their language and make contributions to the document.

In every moment, two versions of the document are released in different languages.

- *Scenario 2*

A teacher creates a document for a university course, and he decides to publish the document in the system. He gives information to the students about how to download the document and how to make contributions to the document.

Some of the students make some contributions to the document and fix some bugs in it. Then the teacher accepts some of these contributions and integrates them into the main version of the document.

The students have the main official version of the document which they can use to make annotations on it. And they can have additional versions modified by them.

- *Scenario 3*

In the University W, the teachers of the subject XXXX publish their documents through the Edukalibre System. The documents contain an amount of posed exercises that have been solved at teaching time.

A student S1 have been including the solved exercises in a local copy of the document of the main branch. On the one hand isn't interesting for the teachers to include these exercises in the main branch of the documentation; on the other hand it's very interesting for the student S1 share its non-official branch in order to obtain the greater amount of solved exercises from other students.

The students S1,S2 and S3 will share an alternative branch with the solved exercises. In this case the student S1 have the edition control of a document and students S1 and S2 will be the contributors.

If student S1 announce its document, others contributors/readers could be added to this non-official-branch.

5. Development process

These is the planification of the phases in the development process:

Phase1

Pre-requisites: No

Description: Testbed with control version system, subversion or gnuarch. The web interface should be available.

Functionality: The testbed with the repository will be installed in the system and a web interface will be available to access the repository.

Phase2

Pre-requisites: Phase1

Description: Scripts to access the repository through ssh.

Functionality: Some scripts will be available in the client part to do the basic actions over ssh to the cvs server, such as create, checkout, commit and update a module.

Phase3

Pre-requisites: Phase2

Description: OpenOffice should be able to interact with the system.

Functionality: A plugin for OpenOffice will be available, which will give the possibility to make some actions in the system such as create a new document, get an existent document and update it.

Phase4

Pre-requisites: Phase3

Description: Colored changes between document versions in OpenOffice.

Functionality: OpenOffice will show the differences between verions of a document in a colored way.

Phase5

Pre-requisites: Phase1

Description: WebDav interface to access the repository.

Functionality: A WebDav interface will be available in the system to access the repository through a WebDav client such as Mozilla, OpenOffice, Nautilus or similar.

Phase6

Pre-requisites: Phase1

Description: Automatic document compilation

Functionality: Some scripts will be available in the server part to process the new versions of the documents to create final formats of these documents (PDF and HTML).