Comparative analysis of collaborative and scalable e-learning methodologies for control theory

Ciro D. Santilli

September 14, 2012

Contents

1 Introduction

The aim of this project is to study current e-learning methodologies, and after deciding which one is more promising, to develop as many features as possible towards the goal on an order of increasing cost/benefit.

The major advantages of elearning are to:

- reduce as much as possible the quality/price ratio of current learning material
- create new experiences for users which were not possible in the usual class-room configuration

Of course, e-learning has some shortcomings, and it is yet an open question whether those will overcome the positive points, and if so, how long will it take.

As the author of this work, I believe that it is only a matter of time before this becomes clear, since in my opinion, only implementing few simple software features will suffice to make e-learning possible, even without taking it into consideration the development of electronic technologies such as higher internet speeds, and e-ind readers which is a potentially slower process than software and methodology development.

After the project requirements are defined, the most promising alternatives to reaching as many those requirements as possible will be studied.

After the alternatives have been analysed, I will implement as much as I can of what I conclude to be the most promising alternative and then present what I have been able to implement.

A part of a short control theory course will be created using the selected methodology in order to show the feasibility of the methods used applied to the teaching of control theory.

2 Proposed schedule

Out of the total 12 weeks left before the final presentation deadline (11/12/2012), of the project, I intend to allocate tasks as follow:

- 1-2 analysis of existing systems.

- 3-10 implementation of as many missing features as possible for the chosen alternative
- 11-12 creation of a control course using the chosen system

3 Requirements

The primary requirements of this system are:

- a way for users to generate textual learning material (referred to simply as "material" from now on), in at least one of the two following formats: pdf or html. Being able to do both formats from a single input format would be very desirable too (but is currently a difficult task).
- a system that allows users to give privileges to others over the internet, in similar fashion to those found in Github or in Joomla. The initial creator of a material has all the privileges, and can choose which privilege will be given to other users. Different privilege levels must exist, being the essential privileges: giving or taking read access to specific users, giving or taking write access to specific users, giving or taking and giving others the ability of giving others read write access to a third user. A level based approach could be used at first, such as read only, read-write access, admin level (can give/take read write access, except of the super admin), and super admin (the only thing he cannot do is to remove privileges from himself).
- a way to view different versions of the material, compare those versions for differences, and find out who modified each version, much as diff linux commands and git blame like commands, such as those used with version control systems such as Git.
- the ability to add metadata to files,specially tag and title. Modifying metadata is also a privilege, which can be given or taken.
- a bugtracker/comments system such as Bugzilla or the built-in Github bugtracker system which allows for doubts and suggestions to be made on specific points of the learning material, and so that all users can view those problems and help resolve them. This system must also allow for a user to ask for (email) notifications from other chosen users. Two

aspects missing on most current bugtrackers and which are essential for this project are:

- the creation of groups, such that a teacher can see and focus on requests from his students before any other requests.
- the ability for users to vote up on not only on issues which they think are important, but also on answers to those issues which they believe to be effective. This can be coupled with importance metrics so that each user can automatically select which issues are more important to him.

Many existing systems are close to fulfilling those requirements, but I believe none yet can fulfil all of them in a convenient manner. A major goal of this project will be to analyse those existing systems, and determine which one would be more profitable to modify to reach the desired requirements faster.

4 Major challenges

- local editing vs. browser editing.

  The positive aspects of local editing methods are:

  - current web browser editing methods are much less advanced than local text editing methods
  - most input formats need to be processed (latex to pdf, or lightweight markup to html + css) before they can be viewed by the adequate viewers (pdf readers, browsers, etc.). More than that, compilation has to be done several times while writing the input file in order to see what is it turning out like. This kind of compilation process, however, is very cumbersome to carry out directly on browsers as of today, and might be costly to carry on a server side.

  The downsides of local methods are:

  - server/local communication is a point which greatly increases the complexity of the project and the learning curve of local methods of development.

  - metadata is meant to be used by the sever, and fits better directly inside a database, forcing users to both commit, then reload the commit web interface, and then finally input the metadata.
  - binary data such as photos cannot be shared amongst users on the server in order to save server resources and avoid redundancy, also making distribution of data easier, such as is the case of Wikipedia which centralises all binary data on Wikimedia. This limitation seems hard to circumvent to me.

- choice of input and output formats. The two major output formats are pdf and html, but if it is not possible to do both which one would should be chosen?

  The advantages of pdf are:

  - mandatory for printing
  - wide range of typesetting possibilities for books such as numbering and bibliography support due to latex, which are somewhat lacking/experimental in html outputs.

  Advantages of html:

  - loads faster
  - additional content such as comments or indexes can be added on the same page as the main content without any difficulty
  - in order to get user feedback, a browser is going to be necessary at some point of any method, so using html encourages users to give feedback

A linked question is which input format should be used, since there are hard to solve conflicts between the specification of a pdf document and an html one.

5 Examined alternatives

5.1 Source code management + bugracker

This combo has already been used for a long time for the development of open source software and has reached great maturity in this domain of application,

however, the techniques used for software should carry to text development since both are nothing but big bunches of organized textual information.

There are however some superficial conflicts that still make this alternative cumbersome for the development of text. I examine here a Git + Github combo:

- the development is local oriented, suffering thus of difficulties of local development such as steeper learning curve and difficulty of sharing binary resources such as images.
- files cannot be modified on a one by one basis, forcing users who want to make merge requests to either pull the whole repository (which may be large, and very large if a repository has a great amount of images), or for the authors to work with one file per repository, which would require creating and pushing to a large number of different repositories.
- merges are line oriented, while the more natural form of text is natural languages modification is word-wise. It is true however that git can do word-diffs via the –word-diff option, but the mergers are still line oriented, and thus inefficient. This could however be circumvented by tweaking git by adding a word-wise commit method, chosen by the user at the moment of creation of a repository, and which cannot be changed afterwards.

I have focused mostly on Github because it seems to be the most complete web interface for git developement, but if this alternative is to persued, it would be necessary to move to another open source git web interface, such as Gitorious. Also in the case of Gitorious for example, the bug tracker is not integrated, so that it would be necessary to choose, tweak and possibly even integrate a separate bug tracker system into it.

5.2 Wikis

Wikis have one major downside: they are not user oriented, that is,

- there is very little little permission management. All users have read write access.

- all users are supposed to work on and reach a consensus on a single concept and a way to present it. Obviously, different people will have different ways to present material, and conflict will happen. Furthermore, it might be good to have different points of view on a single concept since each one might be more adequate for each person.

5.3 Blogs and content management systems (CMS)

By CMS it is meant systems like Wordpress, Joomla or Drupal. Blogs have been grouped with them since they can be seen as nothing but a simple individual oriented CMS, being that some of them such as Wordpress are starting to become more and more complete CMS systems.

TODO

5.4 Connexions

Connexions (http://cnx.org/). While it shares some aspects of CMS, it has such a great emphasis in pdf/html/ TODO