WebTeach: a Cooperative Web Environment for Teaching Science and Technology

Andrea Sterbini
Computer Science Dept., University of Rome 1, Italy
sterbini@dsi.uniroma1.it

Franco Bagnoli
Applied Mathematics Dept., University of Florence, Italy
bagnoli@dma.unifi.it

Abstract

In this paper we illustrate our experience of using the cooperative framework provided by an extended Wiki Web as a tool for distance teaching of Sciences and Technology.

A Wiki Web is a web site in which pages can be directly edited by users through the usual browser interface. A simple syntax allows the user to add contents to the web without knowing HTML. The pages are automatically linked and created whenever a particular pattern is inserted in the text. This completely removes the "one webmaster syndrome" and focuses the users to cooperatively work on the web contents.

To teach Sciences or Technology we have extended a Wiki system with a plugin API and developed several plugins, e.g. to handle LaTeX fragments and plots of 2D/3D functions.

1 Introduction: teaching science by web

The web was born in the physics community in order to ease the spreading of scientific information. One of the biggest advantage of the web in its present form is the (relative) possibility of viewing contents without using a particular software product or operating system. Indeed, it is not unusual in mathematics, physics and other scientific communities that teachers use a different system (some flavor of UNIX or Macintosh) than the vast majority of students. The web has represented a radical revolution in the way information is distributed.

However, after about 20 years since its introduction, no structural changes has occurred in the way information is elaborated: writing for the web is almost the same than writing for paper. The text has to be carefully formatted or converted to a formal language (say, HTML) and users are strictly separated from authors. In particular, collaborative editing requires special tools, often proprietary or not available for all operating systems. Interaction with users (say, collaborators or students) is generally painful, especially for scientific arguments: formulas, scientific symbols and figures are not easily communicated via e-mail or web pages.

On the other hand, the mass usage of electronic media has revealed that a large fraction of communications do not follow the client/server scheme, but belongs to the peer-to-peer class. In particular this applies to e-mail, GSM short messages (SMS) and in general phone calls. All the peer-to-peer communication is characterized by informal formatting.

We started experimenting distance teaching with the following social requirements in mind: a democratic environment, possibly self-organized, and community support for both technical and content aspects.

One of the basis of scientific investigations is democracy: the scientific material is generally made available as soon as possible via preprint servers, data from experiments or source code for simulations is generally available upon request, etc. This approach is also the base of the growing field of free software (see www.gnu.org). We would like to favor the constitution of collaborative environments among teachers and students in which (trusted) collaborators can easily share editing of teaching material.

The free software experience has shown that a *public* project may grow slower than an highly engineered one, but is less prone to "sudden stops" due to exhaustion of developers' resource. Examples are discontinuation of proprietary software products, or elaboration of collaborative material (say, a book) when the organizer is distracted by other projects.

These considerations lead us to the following technical points:

No particular system requirements. We do not want to use proprietary systems not available for some operating systems or that require fees or special software (both from the server and the client point of view). This has the advantage of allowing the technology to be usable by all, including for instance third world countries.

Generic source formats for contents. Each community has a preferred formatting tool: in mathematics for instance almost all material is formatted using LaTeX, while in other disciplines Microsoft Word documents may be acceptable. In our view, converting from a format to another is a computer's, not users', task.

Avoid the one-web-master bottleneck. All kind of web teaching require great efforts to keep them alive: the material has to be checked for consistency and errors, new material has to be added every day, feedback from users should be considered a precious resource and promptly handled. On the other hand, only rarely teachers have at their disposal dedicated people for managing the front-end, and teacher time itself is a widely fluctuating resource. A hierarchical organizations of the work (that e.g. implies the need for web managers collaboration) is much less robust than a diffused organization with respect to this kind of fluctuations.

Our project (*WebTeach*) is composed by three parts: A web-based database of tests (*WebTest*), a manager of multiple choice quizes (*WebCheck*), and a collaborative writing tool (*WebWrite*).

The *WebTest* tool allow teachers to announce tests and students to register for them by web. This relieves teachers and students to be physically present at the university just for bureaucratic reasons. Moreover, in case of sudden changes in test scheduling, students are automatically alerted by e-mail, or, in the near future, by an SMS message on their cell phone. Tests can be used both as a self-assessment tool (by web) or as formal exams, at the university, on paper.

When inserting a new test in the database, teachers have the option of creating a multiple choice quiz (at present the *WebCheck* option is being implemented). The quiz may be handled on paper or by web. In case of traditional paper handling, after the examination the students will be allowed to immediately check their results. The students will send their replies (a string of numbers) by an SMS message to the system from their cell phone (which are widely used by Italian students), or through a web interface. Afterward, they immediately receive a confirmation message with the list of right/wrong answers. In any case, the test results are machine-corrected and the data inserted into a database. This database allows checking for too easy questions or completely missed right answers, which could be an indication of a weak point in teaching. The automatization of quiz management should also favor the continuous monitoring of students' progresses, particularly needed in this moment in which the organization of Italian universities is changing toward more compact modules.

The core of our project is the *WebWrite* tool, which is used to produce documentation for courses, handle Frequently Asked Question (FAQ) threads, populate the database of quizes, and in general may serve as a cooperative distance communication tool in the teacher/students community.

We decided to adopt the WikiWikiWeb [1] point of view. WikiWebs appears as usual web sites in which all pages can be directly edited by users through the web browser itself. A simple syntax is used to add contents to the web without knowing HTML: the approach is to input simple text as one does when writing an e-mail. It is the server's task to present this text in a nice way.

In the Wiki jargon a web page is termed "topic", and a homogeneous set of topic is termed a "web". A topic name is usually distinguished because it is written with uppercase letter in the middle, as for instance "`WebTeach`". The presence of a topic name is automatically recognized, and the system adds either the hyperlink to the corresponding page, or signals the possibility of creating the missing page. This favors a top-down approach to writing: authors start from the index, and then populate the web by clicking on the orphan links.

There are several different implementations of WikiWikiWebs, and also other similar approaches to cooperative environments, either free or proprietary. In the following, we shall use the term *Cooperative Webs* (CoWebs) as a generic name for all kinds of WikiWikiWebs approaches.

CoWebs have been introduced for producing documentation of software products (see for instance `www.zope.org`). Their ability of catalyzing virtual communities promoted their use for social communications (see `www.c2.com`). They have also proved very useful as teaching tools [2].

2 Implementation from the users' point of view

From the beginning we have chosen TWiki (available at `www.twiki.org`), a successful and highly developed Wiki implementation. TWiki has a lot of desirable features.

2.1 Categorization of topics

Each topic can be classified by means of several category indices: lesson, course, prerequisites, related topics. The student can find his/her way through the didactic material both by doing full-text searches or by following the category links.

TWiki allows a very easy definition of new categories: simply edit a particular page and add the new term. A web can contain multiple category schemes applicable to a topic. This allows both a rough form of work-flow and a finer classification of each topic.

2.2 File attachments to topics

All topics can have files attached, i.e. uploaded to the server. This makes simple the distribution of software or handouts. The uploaded files can be linked in the topic text, thus allowing the inclusion of images/multimedia files in the page shown.

2.3 Templates and Skins

TWiki uses a template system to layout all pages in a common and consistent way through each web. Templates can be site-wide or defined on a web-by-web basis. This mechanism also allow the user to select the way s/he wants to see the pages through the choice of an appropriate "skin". Apart for the more playful side of this, skins are used to:

- get a printable version of the page,
- activate/deactivate the view of topic comments (see 4.4),
- use pages in other languages (see 6.6).

2.4 Fine-grained access control managed through the web itself

Normally only the students of the course are allowed write access to the corresponding web. Some topics (the syllabus, or the exam's results) are instead editable only by the teachers.

TWiki allows the definition of access rights both at site-wide or web-wide and at single page level. The administration of the users is easy, because the definition of groups of users is itself stored as a topic (editable only by the superusers group). A hierarchy of groups can be designed by group-inclusion.

To help in checking for proper group definitions we exploit a tool that produces a graphic map of the groups relations (inclusion and ownership) with the help of the `graphviz` package (see at `www.graphviz.org`).

2.5 Automatic email notification of changes

This is a feature that has completely replaced our normal bulletin board. Our students are daily notified of all page changes in their course web. The management of the mailing list is directly done by the students themselves, just by editing the WebNotify page in the course web. This relieves us from the burden of handling the list.

2.6 Full-text searches

As in most portals, TWiki implements a simple full-text search engine with optional regular expressions.

2.7 Version control of topics trough RCS

Mistakes or misuse are avoided by storing all the versions of a TWiki topic. No page can be deleted by the normal user.

3 Implementation from the server's point of view

To be concrete, let us describe the steps followed by the system to present a topic's content:

1. The contents of the topic are stored as a text file, allowing the use of normal UNIX tools like grep to perform searches, RCS for version control and so on. In the source file the formatting elements are kept at minimum: emphasized text is simply surrounded by asterisks or underscores (an e-mail convention), bullet lists are marked by whitespaces followed by an asterisk and a space, URLs are just plainly written, and so on.

   The author is allowed (but discouraged) to use HTML formatting.

2. When visualization is requested, the text is elaborated in order to format it as HTML, inserting bold, italics, bullet lists, hyperlinks, etc. There is the possibility of inserting "special" commands to include other topic, insert the user's name or the date of the day, and so on. In particular, web indices are plain pages containing just a "special" command.

3. The test is then embedded into a template, which furnishes the appropriate skin. This skin generally includes buttons for navigation, searches, editing, etc. Actually, the templates are just pages with several "special" commands, and they can be edited using the same mechanism of other pages.

4. When editing, a simple text area with the source is presented, so that the author is not distracted by formatting tags. While this approach keeps the system requirements minimal (one can use very spartan hardware and software), we are planning a Java interface with WYSIWYG feeling.

The above features make TWiki a powerful content manager, perfect for creating portals, as it combines web, file server and hyper-linked discussions in a simple framework.

But to teach mathematics, physics or other scientific arguments we needed something more than a normal Wiki.

4 TWiki extensions for scientific teaching

First of all, we realized the need of inserting mathematical formulas, using the *normal* LaTeX syntax. We also liked the idea of attaching an existing LaTeX document (or a postscript figure, or a Microsoft Word text, etc.) to a page and include it into the text as is usually done for GIF images. Moreover, we would like to insert plot of functions, without the burden of creating a GIF image and attaching it to the page.

Finally, we considered that most of discussions in Wiki sites assume the form of threads, with questions, comments, answers developing in a hierarchical manner. This is clearly possible using the usual Wiki unstructured editing, but implies a strict discipline, not usual among students. Moreover, we would like to have different permissions for editing a topic and for adding questions or comments to it.

We started modifying the TWiki code (thanks to its free-software approach), but we soon realized that this would imply loosing the contact with the TWiki developers community (`www.twiki.org`), thus introducing a weakness in the support and preventing the possibility of easy upgrades of the software. On the other hand, the rest of TWiki users were not happy of adding heavy and useless (from their point of view) features to the generic TWiki code.

So we decided, in accordance with the TWiki development pool, to design a generic plugin API (which is now part of TWiki) that permits to selectively include new features.

Plugins can be activated on a web-by-web basis. They extend the system by adding functionality in several steps of the page creation: at the beginning (in order to handle the current session), just before rendering the topic, for each line, at the end of the rendering, just before saving, at the end (to handle a possible redirection).

4.1 `LaTeXPlugin`: include LaTeX fragments in the topic text

LaTeX is the *lingua franca* for writing mathematics. Support for formulas in HTML is at the moment rudimentary (if not absent). In a future we will be able to use MathML but for the moment the best way to show math in web pages is either by using translators from LaTeX to HTML (e.g. `TTH`, `latex2html`, `hevea`) or with a Java applet embedded in the page.

To keep the browser's requirements minimal we have chosen to implement a plugin to automatically transform to HTML the LaTeX fragments present in a topic. As the conversion is slow (it produces several small bitmaps) the plugin is activated at save time, so that the view pipeline is not affected by the conversion. The page produced is, anyway, passed through the normal rendering pipeline so that automatic links (topic names, URLs) present in the LaTeX part can be handled by the TWiki machinery.

4.2 `SlurpPlugin`: include and convert attached documents

In the normal TWiki installation, other topics can be included "as-is", or attached GIF images can be inserted using the HTML syntax. We have unified and extended this syntax to allow inclusion and automatic conversion to HTML/GIF/JPEG of an arbitrary attached document (at present only LaTeX, plain text or postscript images). This permits a quick "wikization" of existing documents. The plugin allows also the automatic unpacking of attached compressed archives containing the topic and its related images.

4.3 `GnuplotPlugin`: insert 2D and 3D plots

We have developed a plugin that allows the writer to embed 2D and 3D plots in the topic text. 2D and 3D plots are created with the free software `gnuplot` and shown as embedded pictures in the page. In this case, also, the bitmap generation is done at save time to keep the visualization faster.

4.4 `CommentsPlugin`: do threaded discussions commenting the teacher's handouts

TWiki already allows a shared editing mode for cooperation with students on the course material. In a minimal setting the teacher can leave the course topics editable from all course students except for some special pages s/he wants to keep editable only by the teacher (e.g. syllabus, exams results ...).

To make the cooperation easier, better structured and to allow free and even anonymous comments we have designed a Plugin to add threaded comments to each topic of a web. The comments are kept in a separate web with different authorizations if needed. Each comment is a topic and is related to a topic/comment in the main web or in the comment web. The titles of all comments to a topic can be shown under the topic as an indented tree (like in a newsgroup). The comments view tree can be enabled/disabled through the skins mechanism.

4.5 `CalendarPlugin`: show a calendar of important events

We are using TWiki as a bulletin board for the course. A synthetic reminder of all course events (exams, seminars ...) is shown as a small monthly calendar in our home page. The calendar highlights (and links) all event days listed in the same or a different page. In the near future this plugin will be linked to the *WebTest* part to automatically reflect the course's exams listed by the automatic reservation system.

4.6 `GetAWebPlugin`: download a full course's web

Downloading a full web content (attachments enclosed) is very useful both for backup and for browsing the content offline.

We are extending this plugin to package the HTML rendered pages instead then the source pages. This way the full course material could be browsed offline or stored in a CD-ROM without the need for a local TWiki installation.

4.7 `InterWikiPlugin`: make easy references to external WWW sites

TWiki already automatically links URLs in the text (external links). This leaves open the problem of keeping all links up to date with link changes. This problem can be solved by an indirect redirection: external links are marked with a special syntax similar to internal ones, and the final conversion to the actual link is described on a special page which can be easily maintained. Moreover, URLs do not convey any semantic meaning.

External links are written as `<alias>:<page>` pairs. E.g. `CSDept:Admin` links to the Admin office of the Computer Science Department, while `ISBN:1234567890` refers to the University Bookshop page selling the course textbook.

This way of referring to external sites is common in other Wiki and is named "Interwiki". Our implementation is made through a plugin and adds two important features:

- The associations between aliases and URLs are kept in a topic. This eases its management and makes possible to use different aliases on a web-by-web basis.
- A more general syntax is used for the URL generation rules (e.g. the above mentioned `CSDept:Admin` is transformed to *http://www.dsi.uniroma1.it/Admin/default.htm*). This adds semantic meaning to the external link itself.

4.8 `PerlSamplePluging` and `PrologSamplePlugin`: show "living" examples of Perl and Prolog programs

These plugins evaluate Perl (or Prolog) examples and show side-by-side a table with the source code, the output and the error streams. This is very useful to embed "running" examples in the handouts of programming-related courses.

5 A Survey of preliminary results of cooperation through TWiki

The system is being tested at the Engineering Faculty of the University of Florence (managed by the Department of Applied Mathematics) (`didattica.dma.unifi.it`) and at the Department of Informatics of the First University of Rome (`twiki.dsi.uniroma1.it`)

The *WebTest* tool has been released two years ago, and at present it is being used by more than 30 teachers for about 50 courses, 200 tests (each with an average participation of 25 students), and a total of about 2000 enrolled students. Its use is steadily growing.

The *WebWrite* tool is at present in beta test. It has been used to produce documentation for three courses (informatics, advanced geometry and calculus) and is used as a "virtual lab" of dynamical systems and statistical mechanics.

The *WebCheck* tool is being used by more than 20 teachers, and will be used for performing the entry test to all new students in the Florentine Engineering faculty (about 1000 tests at a time).

6 Near future development

We are continuously improving the system; next step is to formally start a public CVS based project (E.g. on `www.sourceforge.net`) to coordinate the development. We are actually working on the following arguments.

6.1 Guiding students through a learning path

By integrating tests and navigation, we can offer students that make systematic mistakes to be automatically guided to the corresponding pages. Categories can be already used to suggest prerequisites, related topics and topics following in the learning path. A graph tool (based on `graphviz`) will be designed to show the graph of paths through the course. The graph will highlight the topics known (tests passed) and the best topics to study next.

6.2 Peer reviews, topic quality, rewards

A common problem in teaching is students' unwillingness of exposing themselves in comments, questions, and so on. We are using a peer review system to stimulate students to "vote" pages, and studying an automatic rate system to reward student participation.

6.3 In-depth correction of complex exercises

We are extending the `WebCheck` quiz manager to handle the automatic correction of more complex exercises [3]. We want to avoid the possibility that the student just "guesses" the right answer and, at the same time, we want to have a finer understanding of his/her errors. An added (and more important) requisite is that the student should find very very easy to input the solution to the system, so that s/he is not stressed/distracted by this task.

To this aim, we are implementing a system that receives the solution from the student and does several detailed checks to retrieve errors. We are working on type of exercises with solutions represented by data-structures. These can be very easy to communicate by web forms or on especially formatted Optical Mark Reader (OMR) forms. Example of data-structures easily representable on a (web) form are:

- undirected and directed graphs.
- finite state automata,
- truth tables, Programmable Logic Arrays (PLAs), Boolean functions,
- stacks, trees, lists ...

We formulate questions in such a way that the student should reply by first solving the exercise and then by giving the solution; this by filling the corresponding data structure.

The correction machinery compares the properties of the data-structure filled by the student with the properties of the corresponding data structure of a correct solution (if more than one exists) given by the teacher. If one of the solutions matches then the answer is correct, else the system looks for the answer with a minimum number of mismatches.

To explain the errors to the student (and to compute the penalty for errors) the system tries to cover the set of mismatches with the minimum number of "mistake patterns" defined by the teacher. A mistake pattern is a subset of the properties checked. Mistake patterns are used to decouple the error of the student from the data-structure properties (a data structure can be used in very different exercises).

Exercises in several fields of sciences (e.g. "Digital Electronics", "Algorithms and Data Structures") can be automatically corrected this way in a manner very similar to the way the teacher is used to.

6.4 Distributed Wikis

As a more general setting than a single server, we would like to distribute the content over different servers, not always on-line, and transparently manage the updates. This would allow us to design distance learning settings where several schools, not always on-line, share common discussion spaces.

The simplest approach is to keep multiple copies of webs (one for server), all copies being read-only except for a single server. Synchronization of servers content is done through (rsync). A more general approach could use `CVS` or replicated databases for the topic storage layer.

6.5 Distance teaching to jailed students

The Wiki instrument is especially suited to assist teaching to jailed students. They share the same inconvenient of far-distance teaching, without the possibility of consulting teachers and friends at will.

Since the jail administration is highly concerned about security, one has to carefully control the traffic of information, especially the possibility of arbitrary external contact via e-mail, HTTP or other protocol. The solution we are experimenting is based on two distributed Wikis (one internal the jail, properly secured, and one external, connected to Internet) whose content are synchronized at scheduled

times. This, at this moment, implies a rigid separation of writing areas in order to avoid arbitration of changes to files. This constraint will be removed when Distributed Wikis will be available.

Both servers shares the same accounts (for students and teachers). The internal server is accessed by jailed students, and the external server is accessed by teachers and other support people. By removing all unnecessary services but the Wiki one, we hope to meet the security level required by jail administration, still providing a useful tool to students.

6.6 Internationalization

Being in Italy, we need a general support for Internationalization:

- All templates are translated and selected through the skins mechanism.
- Accented chars should be usable in TWiki automatic links.
- Multiple language versions of a topic must be handled/shown.
- When a page in the preferred language is not available a default one could be produced by automatic machine translation (e.g. by using the `babelfish` web service).

6.7 Cache partial results

Our `LaTeXPlugin` and `GnuplotPlugin` cache their bitmaps and try to avoid the recomputation of pictures already stored in cache. From parts of these two Plugins we want to make a general Cache module for TWiki.

6.8 Automatic format conversions

We want to leave the users as much freedom is possible for producing documents to be inserted in Wiki pages. To this aim we are extending the `SlurpPlugin` 4.2 to convert among several formats Word, PDF, XML and HTML. A similar conversion from HTML to PDF or XML would ease the production of books from webs to be printed or stored in CD-ROM.

6.9 Include embedded examples written in other programming languages

The `PrologSamplePlugin` and the `PerlSamplePlugin` share the need to carefully manage the code to be evaluated to avoid security breaches, denial of service attacks and stupid mistakes. We are designing a common framework for code execution within restricted resources.

Moreover, we want to extend the plugins to handle other programming languages: Mathematica, Assembler, C, Pascal, SQL.

6.10 Annotate pictures with drawings, text and URL links

Picture annotation could be effectively used for automatic assessment or for teaching.

A TWiki extension is available to draw simple drawings. This will be extended to produce annotation of images with text, lines, arrows, boxes and WWW links.

References

1. Bo Leuf and Ward Cunningham. “The Wiki Way: quick collaboration on the web”. Addison Wesley. 2001.
2. “A Catalog of CoWeb Uses”. Collaborative Software Lab, College of Computing, Georgia Tech, 2000.
3. Andrea Sterbini. “Automatic In-depth Correction of Exercises”. Abstract published in this Conference.