2012 IEEE Fourth International Conference on Technology for Education

Using Org-mode and Subversion for Managing and Publishing Content in Computer Science courses

[*]Sankalp Khare,[†]Ishan Misra, [‡]Venkatesh Choppella
International Institute of Information Technology, Hyderabad, India
Email: [*]sankalp.khare@research.iiit.ac.in, [†]ishan.misraug08@students.iiit.ac.in, [‡]venkatesh.choppella@iiit.ac.in

Abstract—**Content creation and management is an inevitable part of teaching a course. This paper describes a novel way of handling this problem using Org-mode, a recently created text-based information management tool being used within the Emacs user community. We list certain desirable features, specially for the purposes of computer science courses, such as support for collaborative development and literate programming. We show how Org-mode compares favourably over other approaches like wikis and other content and course management systems. We describe why the combination of Org-mode and version control is suitable for creating and publishing content quickly, with minimum overhead in a collaborative manner.**

Keywords-Teaching, CS Education, Literate Programming, Emacs, Org-mode, Subversion, Content Management, FOSS.

I. Introduction and Motivation

Content management is an integral part of teaching a course. Course content (in the form of notes, presentations, handouts) includes text, source code, images, etc. One also has to record notes on the road-map of the course, i.e. meetings, todos, discussions etc. to help plan the logistics of the course and proceed in an organised manner.

One aspect of Computer Science courses that distinguishes them from others is the presence of program listings and code. Literate programming (LP) [1] is a way of enclosing program fragments in a narrative, rather than including comments within the program. Studies have shown that LP is an effective means of teaching programming to students [2]. Org-mode is one of the few tools available today that allows instructors to leverage LP in teaching programming [3].

In choosing a content management system, the following features merit consideration :

- *Ease of editing* – A minimal overhead writing environment so that the author may concentrate on the content, rather than its styling.
- *Presentability* – The material must be exhibited/published in a way that is aesthetically and otherwise pleasing.
- *Structured content* – The source must be structured and organised.
- *The ability to embed and run source code* – The system must be able to provide support for the *tangle* and *weave* [1] features of LP in a variety of programming languages.
- *Portability of sources* – The material and its development should be portable to all development platforms (operating systems) as well as publishing platforms (wiki, CMS etc).
- *The ability to export to different formats* – The course content should preferably be format agnostic and easily exportable to a variety of formats.
- *Collaborative development* – The people involved in the course (course instructor, teaching assistants) should be able to add/edit material.
- *Reusability and continuous development* – Most courses are offered every year or every semester. The material must, thus, be stored and managed in a way that allows iterative and continuous development over time and encourages reuse of the core content.
- *Access control* – One may need to grant different access levels to different users or groups of users. The system should have support for this.
- *Automation* – All aspects apart from the authoring of the content must be automated. Hence, after the initial setup, subsequent updates must be easy to perform.

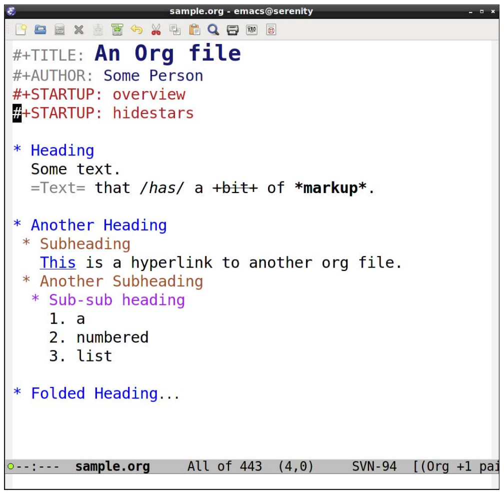
The approach we present leverages the power of FOSS tools to provide a unified solution that satisfies the properties listed. We have been using this approach to manage courses at our University. See [4] and [5] for course websites developed and maintained using the approach we describe.

II. Assessment of Existing Systems and Models

We present our analysis of some commonly used course content management systems in the light of the desired features listed in Section I and a brief comparison with our system in Table I.
Wikis : Wikis are very convenient for entering text and even source code snippets. They do not, however, have support for evaluating code. Content like `PDF` files, images etc. needs to be created using other software. The sources for this content need to be maintained separately.
Content Management Systems : In most CMSes, the content is “locked” in their databases, making it difficult to migrate it to something different (e.g. a wiki). In addition, they suffer from the drawbacks we mentioned for wikis.
Moodle : Moodle is a Learning Management System (LMS) which is designed specifically to let instructors manage course content. While it has been deployed for various courses successfully, it fails to meet the requirements of embedding and running source code, ease of migration, and continuous development. LMSes also use databases and hence suffer from the same problems as CMSes. In Moodle, content is uploaded as individual files. This is inconvenient when the number of files is very large and the files are of different types.

TABLE I
COMPARISON OF SYSTEMS

Feature	**Moodle**	**Wikis**	**CMS**	**Emacs and Org-mode**
Content Portability	No	No	No	Yes
Embed and Run Source Code	No	No	No	Yes
Export to Different Formats	No	No	No	Yes
Version Control with Traceability	Yes	Yes	Yes	Yes
Access Control	Advanced	Advanced	Advanced	Advanced
Learning difficulty	Medium	Low	Medium	Medium
Maintenance of sources for extra content	External	External	External	Internal



```
sample.org - emacs@serenity
#+TITLE: An Org file
#+AUTHOR: Some Person
#+STARTUP: overview
#+STARTUP: hidestars

* Heading
  Some text.
  =Text= that /has/ a +bit+ of *markup*.

* Another Heading
 * Subheading
   This is a hyperlink to another org file.
 * Another Subheading
  * Sub-sub heading
    1. a
    2. numbered
    3. list

* Folded Heading...

o--:---  sample.org     All of 443  (4,0)     SVN-94  [(Org +1 pai
```

Fig. 1. An Org file opened in Emacs

Any system which requires instructors to upload files requires that the source of these files must be maintained separately. All the systems we have mentioned so far have this problem. They are also not suited for portability of source across developing and hosting platforms. None of them provide support for embedding *and* running code (LP).

We use files under version control (not databases) to store our content. Filesystems are built to work across platforms without the user having to write extra code. They are well suited to the task of handling semi-structured data and are very easy to backup. The advantage of using a database over a filesystem is mainly in terms of performance (search and retrieval due to indexing and caching) and reliability in transactions [6]. To manage content for a moderately sized (within 5-10 GB) course, we find these database properties to be an overkill. Using databases is not suitable to the requirements of portability, ease of use and content migration. In order to use databases, one needs to write additional software (queries).

The advantage of version control is that it allows for traceable, collaborative development. The instructor can let other people contribute to the generation of course content and finally all of the content is aggregated in the version control system's repository. We use Subversion [7] but our content management process is versioning system agnostic.

III. EMACS, ORG-MODE AND SUBVERSION

GNU Emacs[1] is the most popular member of the Emacs [8] family of text editors. Emacs has a number of functions and key-bindings which make text editing in Emacs highly efficient.

Org-mode [9] is the mode in Emacs which we use for creating content. Org files have a foldable outline-based document structure. They support text markup, hyperlinking, creation of lists and tables, inclusion and generation of images, export to various formats, and much more[2]. These files are stored as plain text. Figure 1 shows an org file opened in Emacs.

Subversion (SVN) [7] is a cross-platform, open source, centralised version control system. It is highly configurable, reliable and extremely well documented. Org-mode files, being plain-text, are ideal for development under version control [10].

IV. AUTHORING CONTENT

Our system utilises Emacs, Org-mode and Subversion to achieve the desired properties listed in Section I. We use Org-mode to create and organise the content of the course. We find the following features of Org helpful in content creation and management :

Org-mode files are plain-text. This offers a great advantage in terms of portability of the source and the ability to edit using any text editor. Org-mode's built-in lightweight markup language allows for authoring formatted text which is both easy to read and write.

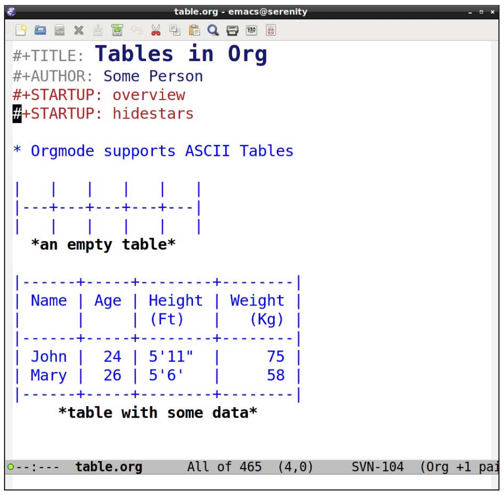
Org encourages the authoring of structured content, with support for nested, foldable sections, cross-referencing, footnotes, automatic table-of-contents generation and much more.

Org has native support for hyperlinks. These may point either to locations on the web or to locally available content – other `org` files, source code, images, documents etc. This allows for the development of modular and organised content.

The built-in ASCII table editor makes working with tables significantly easier and faster in Org-mode as compared to LaTeX, HTML or most other non-WYSIWYG document authoring systems. Figure 2 shows a sample table.

Babel is an Org-mode feature that allows one to include code fragments inside the org file. These code fragments can then be evaluated and their results embedded into the output. It includes support for various languages like Python, R, C, Lisp etc. [11]. We also generate flowcharts/graphs by embedding `ditaa` (Figures 3 and 4) or `dot` code. This particular feature is very useful since it not only allows ease of embedding code, sample outputs etc. but also ensures that the published code is correct.

[1]All instances of the term "Emacs" in this paper refer to GNU Emacs.
[2]This paper has entirely been written using Org-mode.



```
table.org - emacs@serenity
#+TITLE: Tables in Org
#+AUTHOR: Some Person
#+STARTUP: overview
#+STARTUP: hidestars

* Orgmode supports ASCII Tables

|   |   |   |   |   |
|---+---+---+---+---|
|   |   |   |   |   |
  *an empty table*

|------+-----+--------+--------|
| Name | Age | Height | Weight |
|      |     |  (Ft)  |   (Kg) |
|------+-----+--------+--------|
| John |  24 | 5'11"  |     75 |
| Mary |  26 | 5'6'   |     58 |
|------+-----+--------+--------|
     *table with some data*

--:---  table.org     All of 465  (4,0)     SVN-104   (Org +1 pai
```

Fig. 2. Tables in Org-mode

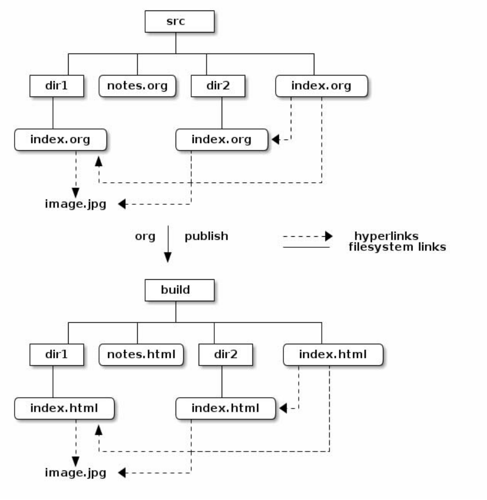


Fig. 3. Publishing with `org-publish`

V. Content Organisation

We describe here the layout we use to keep our content in an organised form, readily usable for publishing, as described in Section VI.

Course content primarily consists of

1) Content intended for students' reference and
2) Content visible to the instructors only.

The first category includes all material which gets published, such as class notes, assignments, reference material etc. The second category consists of metadata related to the course, like notes made during instructor meetings, task distribution related information, accounting of students' scores in various areas, ongoing work on exam drafts and other such material.

Course content for each course offered is maintained in a separate SVN repository. Here is the top-level view of a sample instance :

```
course-svn
`-- offerings/
    |-- 2011-spring/
    `-- 2012-spring/
        |-- admin/
        |-- build/
        |-- elisp/
        |-- makefile*
        |-- org-templates/
        `-- src/
```

The directories `2011-spring` and `2012-spring` respectively store the course material for two semesters when the course was offered. The contents of the `2012-spring` directory serve the following purpose :

- `admin` : Contains course metadata (not published).
- `build` : Target directory for the output.
- `elisp` : Emacs Lisp files used in publishing.
- `makefile` : Provides commands for publishing.
- `org-templates` : Templates for the org files.
- `src` : Contains all the course content.

This houses the source which is used to generate the course website. It comprises of numerous org files spread over many directories and sub-directories, with hyperlinks referencing files from within them.

VI. Publishing Content

Content is published to XHTML using Org-mode's `org-publish` feature. Our publishing process is based upon the tutorial on the Org-mode website [12].

A. The Publish Step

`org-publish` processes a directory tree consisting of related `org` files and generates an identical directory tree with corresponding `html` files, as shown in Figure 3. All org-hyperlinks are converted into HTML hyperlinks. Attachments and other static content gets copied into the corresponding destination directories.

In our setup, the export process operates upon the `src` directory and writes the resulting website into the `build` directory. This local copy can then be inspected to ensure that it is up to expectations.

The commands to publish content look like this :

```
$ cd course-svn/offerings/2012-spring/
$ emacs --script elisp/publish.el
```

B. The Export Step

Once published, the generated content can be *exported* to a server that hosts it on the web. The export command is typically an invocation of a tool like `scp` or `rsync` which transfers the local copy to the web hosting server.

C. The Makefile

We use a `makefile` for the publish/export commands, making the process more automated and modular. The makefile can have additional targets to remove the locally published website, run an svn update and so on.

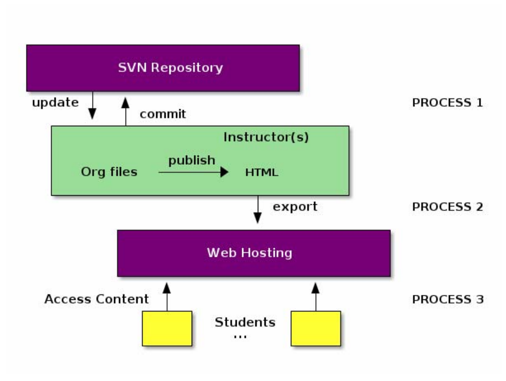


Fig. 4. Workflow of our process

D. *The Content Management Workflow*

The content management process we use can be divided into three asynchronous processes (Figure 4).

1) Instructors commit to the repository and update their copy of the source with changes made by other contributors.
2) Published content is transferred from an instructor's machine to the hosting space.
3) Students browse the published content.

VII. Task management in a course

A. *Generating Exams*

Using Org-mode's commented headings feature, both questions and solutions can exist in the same file. All content under a commented heading is ignored while publishing the org file. Once an exam is over, all that needs to be done is to uncomment headings that contain the answers so that they get included in the published output.

B. *Management of tasks using TODOs, checklists*

We use org for assigning and managing tasks. A `TODO` can be tagged with the designated person's name. We also use the checklist feature to keep count of how many tasks have been completed from a given list. This is useful while making minutes of a meeting and tracking task progress.

C. *Spreadsheets for marks*

Org allows us to make spreadsheets out of tables by using formulae in table cells. We use these for storing marks and computing various statistics for the purpose of grading.

VIII. Challenges and Limitations

Our approach works best for an instructor who manages his own course content. It does not aspire to be a method for large scale management of a number of courses. This is not a problem in a typical scenario where an instructor offers a handful of courses per year. It must also be noted that our approach generates static content only.

IX. Conclusion

The approach presented in this paper opens up a wide array of possibilities for teaching CS. One particular area where this approach truly outshines existing methods is its ability to incorporate Literate Programming seamlessly. It is flexible enough to accommodate the requirements of different instructors and can be adapted to meet them.

Learning Emacs, Org-mode and Version Control may seem like bottlenecks in adopting this approach. We have, however, taught these technologies in a freshman undergraduate course with about 200 students. The students took merely 3 weeks to become reasonably proficient in them. This should bolster confidence in fresh adopters of this approach.

References

[1] D. E. Knuth, "Literate programming," *The Computer Journal*, vol. 27, pp. 97–111, May 1984.
[2] B. Childs, D. Dunn, and W. Lively, "Teaching CS/1 Courses in a Literate Manner."
[3] E. Schulte, D. Davison, T. Dye, and C. Dominik, "A multi-language computing environment for literate programming and reproducible research," *Journal of Statistical Software*, vol. 46, no. 3, pp. 1–24, 1 2012.
[4] "Principles of Programming Languages, Monsoon 2011, IIIT Hyderabad," http://pascal.iiit.ac.in/ popl, 2011.
[5] "IT Workshop 2, Spring 2012, IIIT Hyderabad," http://pascal.iiit.ac.in/ itws2, 2012.
[6] J. Gray and A. Reuter, *Transaction Processing: Concepts and Techniques*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1992.
[7] "The Apache Subversion Software Project." [Online]. Available: http://subversion.apache.org/
[8] R. M. Stallman, "Emacs the extensible, customizable self-documenting display editor," *ACM SIGPLAN Notices*, vol. 16, no. 6, pp. 147–156, 1981.
[9] C. Dominik, *The Org-Mode 7 Reference Manual: Organize Your Life with GNU Emacs*. UK: Network Theory, 2010, with contributions by David O'Toole, Bastien Guerry, Philip Rooke, Dan Davison, Eric Schulte, and Thomas Dye.
[10] I. Barton, "Using version Control with Your org files." [Online]. Available: http://orgmode.org/worg/org-tutorials/org-vcs.html
[11] E. Schulte and D. Davison, "Active documents with org-mode," *Computing in Science Engineering*, vol. 13, no. 3, pp. 66 –73, may-june 2011.
[12] S. Rose, "Publishing Org-mode files to HTML." [Online]. Available: http://orgmode.org/worg/org-tutorials/org-publish-html-tutorial.html