Australasian Journal of Educational Technology

2009, 25(4), 459-470

Development of course material in a multi-author environment

Michael Schlotter
University of Bath

Software for text processing and presentation design is becoming increasingly sophisticated. Nevertheless, it is difficult to find a good solution for collaborative writing of technical course material, allowing the creation of high quality lecture notes and presentation slides from a single source. This article presents a new editing framework for the development of continuing professional development (CPD) teaching material. The courses are developed and delivered jointly by a team of academics, and comprise various engineering subjects. The described workflows, however, are equally applicable for any higher education course. The needs of authors are explored, and good teaching and learning practices are established. Based on these investigations, a catalogue of software requirements is created and compared with features of current commercial and open source applications. Despite the general trend to use graphical interfaces, it is concluded that the workflow has to be based on source files written in a markup language in order to fulfil all specifications. Although the Extensible Markup Language (XML) has become increasingly popular during the last decade, a LaTeX based solution is preferred. Reasons for that decision are discussed, and the developed editing framework is described in detail.

Introduction

The demand of the UK industry for highly qualified engineers is high, while the supply of graduates is in decline (Spinks, Silburn & Birchall 2006; Hodgson, Farr & Gindy, 2004). As employers struggle to recruit personnel with the required knowledge, they may either select less suitable candidates, who will need further training, or improve the skills of their existing workforce. Continuing professional development (CPD) courses are ideally placed to suit the industry's education needs and allow employees to enhance their career prospects (Paton, 2002). The Centre for Power Transmission and Motion Control (PTMC) at the University of Bath has been offering CPD courses on fluid power systems and related subjects for over 25 years. They are popular with delegates from various industrial sectors, and they also form part of the centre's postgraduate education and research programs.

Courses are usually delivered in 4-day blocks during the academic term. Students who want a formal qualification can obtain a Postgraduate Certificate, Diploma, or MSc by attending multiple modules and fulfilling certain assessment criteria. Lectures are presented by various academics in the department, and tutorials, design exercises and laboratory sessions are run by research officers and technical staff. Five different courses are offered regularly at the university, and one course can be completed via distance learning. Occasionally, special courses are delivered, which are tailored to the needs of a particular company. The focus and contents of the standard modules has evolved steadily over the years. When the education requirements of the industrial customers change, or interests and backgrounds of teaching staff varies, the lecture material is adapted accordingly.

The last major review took place in the late 1990s, when all electronic documents were converted to Microsoft *Office* format. The notes were divided into chapters, which usually correspond to a 1-hour lecture. Appropriate presentations were created in *PowerPoint*. Since this review, individual members of staff have edited material for their lectures independently. Before each course, lecture note masters were updated and photocopied, and electronic presentation, if available, collected. This was a labour intensive process, as the electronic documents were scattered over multiple PCs. It is easy to imagine that the overall quality of the courses has not necessarily improved over the years, mainly because individual teaching staff did not know what the others were doing. Repetitions of material in multiple lectures, or presentation in an illogical order became increasingly common problems. Furthermore, the visual appearance of notes and presentations became unsatisfactory due to a lack of strict guidelines and software limitations.

In 2007, it was decided to revise all courses and update the lecture material taking into account the experience gained in the past. Particular focus was put on developing a new document creation and management strategy, which is suitable for a multi-author environment. The following sections explore the demands of authors and students, examine various possible solutions, and describe the developed editing framework, which is released as free software under the GNU General Public Licence (FSF, 2007).

Challenges

The challenges encountered when developing CPD course material are similar to those for standard undergraduate lectures, however, some important differences have to be considered. First, course participants come from a variety of backgrounds - PhD/MSc students, recent graduates, experienced industrial delegates, etc. Although this mix is considered to be beneficial for mutual learning, the different expectations and abilities of these groups need to be taken into account. Second, lectures are delivered in block format rather than over the course of a semester. There is therefore little time for self study and student-teacher interaction between lectures. Finally, the material is written by multiple authors, who may not necessarily present the information. It shall now be explored, how these problems can be addressed, and how they translate into demands on the authoring environment.

Fulfil the expectations of participants

Observations in the past have shown that the industrial delegates' satisfaction with courses is highly influenced not only by the course content but also by the quality of lecture notes and presentations. Important measures are correctness, clarity, completeness, and a professional visual appearance.

The notes support the presentations and serve as an important reference during tutorials and design exercises. Most delegates use them to solve real problems in the workplace, too. In order to enable quick access to the required information, extensive cross-referencing and automatic generation of a document index should be supported by the typesetting system.

Feedback collected over many years suggests that most participants are sequential learners, who prefer to see direct problem solving with detailed explanation of individual steps (Felder & Silverman, 1988). They become dissatisfied if the lecture notes do not follow the slides in the exact order; a situation that commonly occurs when lecturers update presentations without changing the notes, or *vice versa*. It is therefore desirable, to generate all material from the same source, so that inconsistencies are avoided.

Kiewra, DuBois, Christian and McShane (1988) have shown that students perform better if they are taking notes. However, Locke (1977) observed that students fail to record a significant proportion of important information, especially if delivered verbally. Considering that CPD courses are targeted towards engineering professionals who often have little practice in note taking, it appears advantageous to hand out a complete text instead of just lecture outlines or copies of slides.

Visually attractive course material can enhance the learning experience, but does not guarantee a good lecture. Research has shown that well designed electronic presentations increase the students' performance (Lowry, 1999) and enhance their overall perception of the course (Apperson, Laws & Scepansky, 2004). Graphics and animations can support understanding, but only if they are relevant (Bartsch & Cobern, 2003; Tversky, Bauer-Morrison & Betrancourt, 2002). These findings imply that the typesetting program should produce a clean, consistent layout, and enable the inclusion of graphics and animations, but discourage authors from adding superfluous effects.

Make it easy to follow

CPD courses are very compact and deliver a lot of information in a short time. An average 4-day course consist of 12 hours of lectures, 8 hours of laboratory sessions, and 8 hours of tutorials and design exercises. It is crucial that students do not "get lost" and struggle to keep up during the course.

If notes contain references to slide numbers, certain passages can be located more easily. This encourages students to make annotations, and stops them searching while they should be listening. Presentation slides on the other hand, should show an outline corresponding to sections in the notes, indicating clearly what has been talked about already, and what will be addressed shortly. Outlines also help the speakers to pace themselves. Because different topics are presented by different lecturers, it is not possible to start with the next chapter if under running, or finish the lecture on the next day if time runs out.

Especially in technical subjects, short animations, which visualise mechanisms, machine operations, or graphical results, are a good way to provide respite from long mathematical derivations. Interactive simulations also encourage discussion between teacher and students, and can be used to prepare for upcoming laboratory sessions. A facility to include such material in presentations would be beneficial.

Allow collaborative editing

Posner and Baecker (1992) created a taxonomy of joint writing with four components: the roles of group members, their activities, document control methods, and writing strategies. They have found that collaboration is most effective if each person works on different parts of the document(s), of which he/she is in full control. A similar strategy shall be adopted here: Each presenter is responsible for his or her lectures, i.e. for the according chapters in the notes and the presentations. However, all members of staff should also be able to correct mistakes, and make improvements to somebody else's material. A project leader plans and oversees the development, and compiles the final documents before each course.

Although numerous tools for computer supported collaborative work have been developed, Noël and Robert (2004) found that groupware tools are not used in most projects due to complexity, unfamiliar working environment, installation issues, or simply because authors prefer writing in isolation. Nevertheless, most participants in their empirical study believed that collaborative writing leads to a better product. When asked what software functionality is particularly helpful, the most popular answer was being able to see changes made to a document, followed by version control, adding comments, identifying who wrote what, and synchronous access to files.

Most investigated projects were short to medium term (a few days to one year) and resulted in one final document. CPD course development, however, is a continuous process, stretching over many years and involving dozens of documents and a varying group of editors. All documents should be compatible and feature the same style, so that lectures can be moved from one course to another. These points translate into additional requirements on the editing software: long term readability and compatibility of source files, support for structured editing, and prevention of ad hoc formatting.

Implementation

The main decision at the beginning of the project was on the editing software. A large number of commercial and open source applications for text editing are available, and finding an optimal solution for a long term project has proven difficult.

Program options

Popular office suites like Microsoft *Office, Open Office*, etc, were quickly dismissed, because they failed to meet most requirements described in the previous section: no global uniform style for all documents, separate creation of notes and presentations, poor support for mathematical content, no advanced page layout functionality like intelligently placed floats, limited cross-referencing between files, only rudimentary revision control and collaborative editing mechanisms, and regularly changing file formats. Desktop publishing applications like Adobe *InDesign* or Quark *XPress* are not intended for long technical publications and focus on page design rather than content development.

Adobe *FrameMaker* supports writing in an XML based workflow, and is the most promising of all commercial applications. It encourages structured authoring, has good support for maths, and allows single source variations of the same document for different purposes, including conditional output functionality. It was not considered further due to the proprietary file formats, lack of support for Mac OSX and Linux, and the relatively high cost.

Open source solutions based on the XML framework include *Docbook* (Walsh, 1999) and the *eLesson Markup Language*, or *eLML* (Fisler & Bleisch, 2006). Both are specialised markup languages defined by XML schemas (*XML-Schema* and *Relax NG*, respectively). *Docbook* was originally developed for writing technical documents related to computer hardware and software, whereas eLML is an environment for creating online and distance learning material. Using the *Extensible Stylesheet Language* (XSL), source files can be transformed into a variety of output formats including (X)HTML for online viewing, PDF for printing, or LaTeX source code for high quality typesetting (Lamport, 1994). *eLML* allows creation of packages for integration with learning management systems (LMS) like *Moodle* and *WebCT*. Typical XML workflows are shown in Figure 1. The main shortcoming of *DocBook*, *eLML*, and similar projects is the amount of work required to create or customise XSL transformations and the handling of maths. Although equations can be included in XML documents with the *Mathematical Markup Language* (MathML), it is too verbose for manual editing.

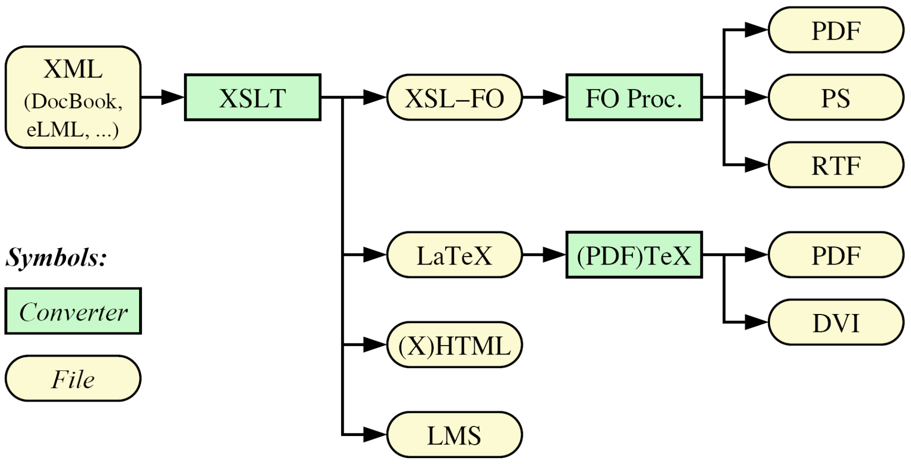


Figure 1: Typical XML workflows for creating documents in various output formats from a single source file

An interesting XML based alternative is the *Integrated Content Environment*, ICE (Sefton, 2006). It is a web based content management application for print and online output, which uses the *OpenDocument Format* (ODF) as storage format. Hence, authors can develop material with the familiar MS *Word* or OOo *Writer* applications. By providing templates, guidelines, and training for authors, as well as tools for version control and collaborative editing, some of the inherent disadvantages of standard word processors are overcome with this approach. Similar to other authoring environments for distance and e-learning, it is targeted towards publishing websites and printed notes. Presentations for face to face lectures are not supported well, mainly because XML separates content and layout completely. Creating sophisticated slides, however, involves careful page-wise layout of content, which is very difficult to automate.

Considering that there is no immediate need for online content, the proposed solution was to skip the XML stage and start with LaTeX sources. LaTeX is a page layout language, which mixes semantic and formatting tags. This allows fine tuning the design while still maintaining the advantages of structured editing. In the past, LaTeX was known for exceptional typesetting quality of printed publications only. This has changed over the last few years, when PDF started to replace the legacy DVI output format. Nowadays, all functionality offered by the PDF format can be accessed with LaTeX, including hyperlinks, interactive forms, advanced page transition effects, and embedded animations and movies, making it suitable for the creation of screen presentations.

All LaTeX sources and macros are simple text files, which guarantees long term readability. Because of the huge number of existing documents, the community generally aims to make new versions of the software backwards compatible. Furthermore, converters exist to translate source files to XML based formats with minimum losses, so LaTeX can be considered as a safe preservation format (Barnes, 2007).

Development of the LaTeX based framework

Every LaTeX document is based on a document class specifying the basic layout. The functionality of the class can be extended or modified by loading additional packages. In the implementation described here, a single source file corresponding to one lecture contains the text and references to graphical content. It is compiled by means of two wrapper files, which load different LaTeX classes for presentations and notes – one loads the *Beamer* class (Tantau, 2007) for creating the presentation, the other one loads the *KOMAScript* report class (Kohm & Morawski, 2008) for creating a chapter of the notes. Additionally, the wrapper files import a package called *ptmcbeamer*, which contains global formatting instructions for both document types. This workflow is illustrated in Figure 2. Before each course, the project leader reviews the relevant material and compiles the lecture notes. As the notes consist of multiple chapters, another wrapper file is used, which can import an arbitrary number of source files, as shown in Figure 3.

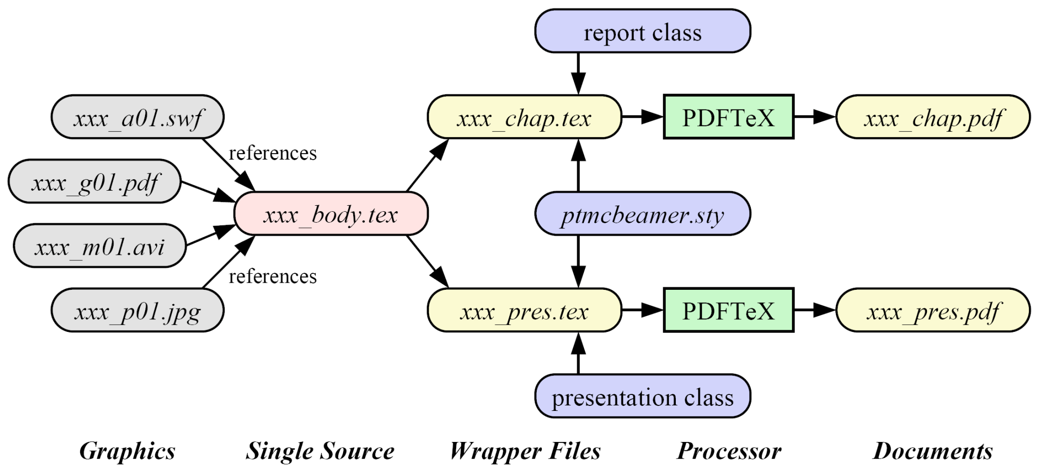


Figure 2: Creation of a presentation and a chapter in the course notes for an individual lecture

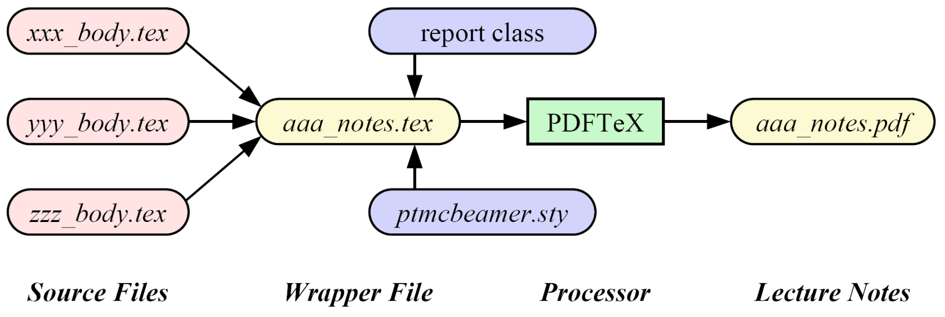


Figure 3: Compilation of lecture notes for a course

Setting up a collaborative editing environment is trivial, because LaTeX source files are plain text. Therefore, standard version control applications can be used to provide advanced functionality such as synchronous editing, tracking changes, merging, tagging, and branching. Subversion is a particularly suitable revision control system, as it can handle text and binary files. It also allows renaming, and provides some advanced functionality by property metadata support. Access rights to the central repository can be assigned based on the roles of the individual project members, eg full access for authors, read only access for all staff in the department, etc.

Very strict naming conventions have been established to handle the large number of source files. All files for one lecture are stored in a directory with a name derived from the lecture title. This 8-character base name is the first part of all files corresponding to that lecture. The second part of each file name is derived from the file content. Although this strategy seems to be very restrictive, it guarantees compatibility on every modern computer platform and avoids confusion due to different file naming habits of the authors.

Possible workflows

Many academics in engineering and science have written some technical articles with LaTeX, so they only need to learn very few additional commands in order to create lecture notes and presentations. The most important concept is frame environments. Everything between the tags \begin{frame} and \end{frame} is typeset on a single slide and in the notes; everything outside frame environments is typeset in the lecture notes only. In addition, the two commands \onlyartcl{...} and \onlypres{...} output the material enclosed by the curly brackets only if notes or presentations are typeset, respectively. The material can be anything from a single character to multiple pages of text and graphics. Almost arbitrarily complex conditional output can be produced with these constructs, but it is advisable to use them sensibly, to keep the structure simple.

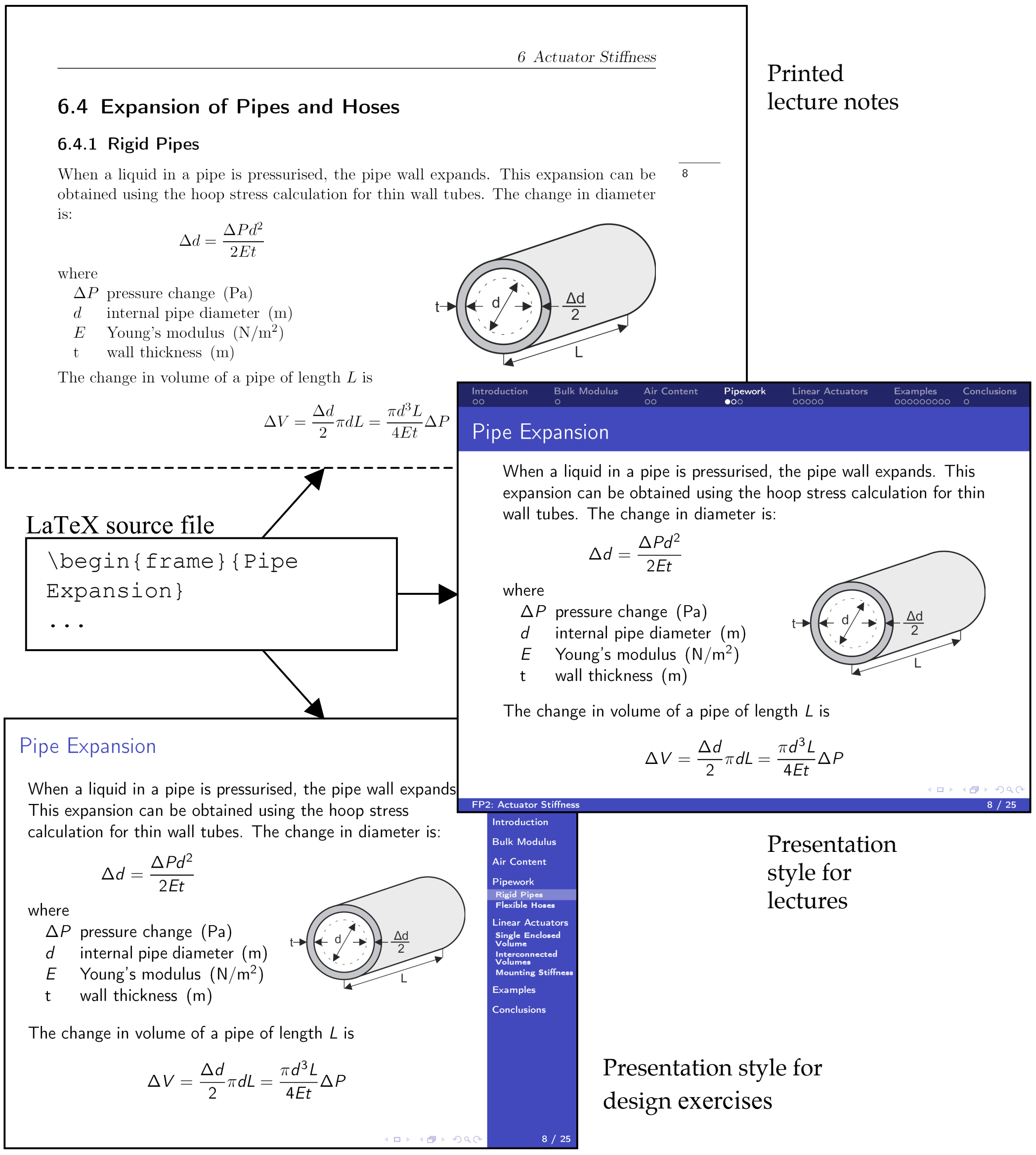
Occasionally, there is a trade-off between creating complex conditional output or simply writing separate segments for different output media. The general guideline is to avoid retyping equations, tables, and other complex content, while the text for notes and presentations may differ. For over 30 lectures created to date, it is estimated that approximately 5% of all content has been written separately for notes and presentations.

Two different workflows can be used to create lecture material: write the chapter and then add frame environments afterwards, or create the presentation slides first and then extend the content for the notes. The method chosen mainly depends on the existing material. In general, it is possible to transform existing files written in a word processor to LaTeX, in which case the first option is most efficient. If only transparencies or outlines are available, the second way may be preferred.

Layout

Figure 4 shows the design of the printed lecture notes and the presentation slides. The notes feature a simple 1-column page design to avoid the need for any manual layout instructions. Table of contents, keyword index, and cross-references are generated automatically, and are implemented as hyperlinks in the PDF file. Small numbers in the outer margin refer to the slide numbers. They help students to follow the lecture, and also facilitate navigation for authors.

Separate presentation styles were developed for lectures and tutorials. The former maximises the available space on each slide. It features a navigation bar at the top showing section titles and the number of slides in each section as small bullet points. The current position is highlighted, which helps to estimate the remaining time of the lecture. All navigational elements are implemented as hyperlinks, enabling jumps to different sections without scrolling. The slide design for tutorials puts even more emphasis on structure. During long mathematical derivations it is helpful to know which steps are currently being performed and which are still to follow. This is achieved by displaying a navigation bar on the right showing all section and subsection titles. No distinct title bar or footer is displayed, because calculations often flow from one slide to the next.



Printed lecture notes
LaTeX source file
Presentation style for lectures
Presentation style for design exercises
t
d
$\frac{\Delta d}{2}$
L

6 Actuator Stiffness

6.4 Expansion of Pipes and Hoses

6.4.1 Rigid Pipes

When a liquid in a pipe is pressurised, the pipe wall expands. This expansion can be obtained using the hoop stress calculation for thin wall tubes. The change in diameter is:

$$\Delta d = \frac{\Delta P d^2}{2Et}$$

where

ΔP	pressure change (Pa)
d	internal pipe diameter (m)
E	Young's modulus (N/m^2)
t	wall thickness (m)

The change in volume of a pipe of length L is

$$\Delta V = \frac{\Delta d}{2}\pi d L = \frac{\pi d^3 L}{4Et}\Delta P$$

8

```
\begin{frame}{Pipe
Expansion}
...
```

Introduction Bulk Modulus Air Content Pipework Linear Actuators Examples Conclusions

Pipe Expansion

When a liquid in a pipe is pressurised, the pipe wall expands. This expansion can be obtained using the hoop stress calculation for thin wall tubes. The change in diameter is:

$$\Delta d = \frac{\Delta P d^2}{2Et}$$

where

ΔP	pressure change (Pa)
d	internal pipe diameter (m)
E	Young's modulus (N/m^2)
t	wall thickness (m)

The change in volume of a pipe of length L is

$$\Delta V = \frac{\Delta d}{2}\pi d L = \frac{\pi d^3 L}{4Et}\Delta P$$

FP2: Actuator Stiffness 8 / 25

Pipe Expansion

When a liquid in a pipe is pressurised, the pipe wall expands
This expansion can be obtained using the hoop stress calculation for thin wall tubes. The change in diameter is:

$$\Delta d = \frac{\Delta P d^2}{2Et}$$

where

ΔP	pressure change (Pa)
d	internal pipe diameter (m)
E	Young's modulus (N/m^2)
t	wall thickness (m)

The change in volume of a pipe of length L is

$$\Delta V = \frac{\Delta d}{2}\pi d L = \frac{\pi d^3 L}{4Et}\Delta P$$

Introduction
Bulk Modulus
Air Content
Pipework
- Rigid Pipes
- Flexible Hoses

Linear Actuators
- Single Enclosed Volume
- Interconnected Volumes
- Mounting Stiffness

Examples
Conclusions

8 / 25

Figure 4: Styles of the printed lecture notes and presentations for lectures and design exercises

Uptake and feedback

Certain difficulties must be expected whenever authors have to adapt their workflow to a new system. In this case, the transition from graphical word processor editing to text based “programming” is a significant change. In engineering and science departments, many academics have some previous knowledge of LaTeX, so that the acceptance of the new system was good. “A significant improvement, especially for mathematical content”, “good system to re-use my LaTeX documents from undergraduate courses”, and “much better consistency” were some of the positive comments by authors. However, some who have no previous experience with this typesetting system complained about the complexity, the need to compile documents before the final output is visible, and the lack of user-friendly editors. These points need to be addressed, and so called 'what-you-see-is-what-you-mean' interfaces such as *LyX* are being evaluated with the aim to reduce the perceived complexity for novice users.

The feedback received from students after the first four courses with new notes and presentations was very positive. Some even asked about the authoring system and where to get it from. Comments included: “excellent visual appearance”, “slide references in notes help a lot during the lectures”, and “very good course material”. The only complaint was that some of the slides look a bit “dry” and should be improved with more pictures. This, of course, can be rectified easily.

Future plans

The development of the editing environment is not finished. Further developments include an optional, more user friendly interface for authors. It is also planned to utilise an automated build tool similar to GNU *Make* or Apache *ANT* for the document compilation process. This will make the current wrapper files obsolete, and allows better control over class options and included packages.

Currently, printed notes are handed out to students in book form at the beginning of each course, and presentations are not distributed. Due to the growing demand for e-learning, however, ways for the creation of online material need to be explored. It is possible to compile sophisticated web pages from a LaTeX source using converters, which either transform the LaTeX source file directly (e.g. *LaTeX2HTML*, *TtH*, *Hevea*) or derives them from a DVI file (*Tex4ht*). Similar to XML based solutions discussed above, mathematical equations pose problems, as they are difficult to describe in HTML. Some of the converters include bitmaps, some use regular fonts and tables, and some can produce *MathML* output. With increasing support of *MathML* by most web browsers, this will probably be the preferred option. If this approach works, distance learning courses can be developed in parallel with standard courses, and authors who are already familiar with LaTeX do not have to learn another markup language.

Flash based animations are already being used for visualising complex relations and computer based tutorials. In future it is planned to make extended use of *Flash* for interactive content, forms for online assessment with immediate feedback, and short quizzes during lectures.

Conclusions

Collaborative development of higher education courses poses a number of challenges. At the writing stage, software should allow the authors to concentrate on the content rather than worrying about the design. Although various applications with a seemingly user friendly graphical interface exist, it has been shown that desired features such as revision control, uniform design, structured editing, single-source creation of notes and presentations, are best supported by workflows based on source files written in a markup language. A LaTeX based framework has been developed, which fulfils all criteria. It was preferred over more modern XML based solutions, because of the authors' past experience, easier implementation, and better manual page layout functionality.

Ultimately, course material has to benefit the students. The current system automatically produces extensive cross-references and navigation aids, including navigation bars on slides, references to slide numbers in the notes, and keyword indices. Inconsistencies between presentations and notes are avoided by the single-source strategy, while the structured editing approach with global design templates guarantees a uniform layout. The PDF output format allows graphical material and animations based on *Flash* technology to be included, which is helpful for visualising complex technical content.

With the central Subversion file repository, the management of the development process became a lot easier compared to the old workflow involving paper master copies of notes and scattered presentation files. Tracking and merging changes, tagging versions, and compiling the latest revision of course material with all corrections is trivial. The only drawback discovered so far is the relatively steep learning curve for new project members who are unfamiliar with the technology. However, the increased effort at the beginning is quickly compensated by the timesaving later on.

References

Apperson, J. M., Laws, E. L. & Scepansky, J. A. (2004). The impact of presentation graphics on student's experience in the classroom. *Computers & Education*, 47, 116-126.

Bartsch, R. A. & Cobern K. M. (2003). Effectiveness of *PowerPoint* presentations in lectures. *Computers & Education*, 41(1), 77-86.

Barnes, I. (2007). The digital scholar's workbench. *Proceedings 11th International Conference on Electronic Publishing*. Vienna, Austria. [verified 25 Jul 2009] http://hdl.handle.net/1885/46750

Felder, R. M. & Silverman, L. K. (1988). Learning and teaching styles in engineering education. *Engineering Education*, 78(7), 674-681.

Fisler, J. & Bleisch, S. (2006). eLML, the eLesson Markup Language: Developing sustainable e-learning content using an open source XML framework. *WEBIST 2006 - International Conference on Web Information Systems and Technologies*. Setubal, Portugal. [verified 26 Jul 2009] http://www.elml.ch/website/en/download/publications/WEBIST2006_eLML.pdf

FSF (2007). GNU General Public Licence Version 3. Free Software Foundation. [viewed 6 Jul 2009] http://www.gnu.org/copyleft/gpl.html

Hodgson, A., Farr. R. & Gindy, N. (2004). Going, going, gone... the engineers of tomorrow. *Engineering Management,* 14(2), 24-27.

Kiewra, K. A., DuBois, N. F., Christian, D. & McShane A. (1988). Providing study notes: Comparison of three types of notes for review. *Journal of Educational Psychology,* 80(4), 595-597.

Kohm, M. & Morawski, J. U. (2008). *KOMA-Script*. Lehmanns, 3rd Edition.

Lamport, L. (1994). *LaTeX: A document preparation system*. Addison-Wesley, 2nd Edition.

Locke, E. A. (1977). An empirical study of lecture note taking among college students. *The Journal of Educational Research*, 77, 93-99.

Lowry, R. B. (1999). Electronic presentation of lectures - effect upon student performance. *University Chemistry Education*, 3(1), 18-21.

Noël, S. & Robert, J-M. (2004). Empirical study on collaborative writing: What do co-authors do, use, and like? *Computer Supported Collaborative Work*, 13(1), 63-68.

Paton, A. E. (2002). What industry needs from universities for engineering continuing education. *IEEE Transactions on Education*, 45(1), 7-9.

Posner, I. R. & Baecker, R. M. (1992). How people write together. *Proceedings 25th International Conference on System Science*, Vol 4, 127-138.

Sefton, P. (2006). The integrated content environment. *Proceedings AusWeb 2006*. Noosa, Queensland, Australia. http://ausweb.scu.edu.au/aw06/papers/refereed/sefton/paper.html

Spinks, N., Silburn, N. & Birchall, D. (2006). *Educating engineers for the 21st century: The industry view.* Royal Academy of Engineering. [viewed 6 Jul 2009]: http://www.raeng.org.uk/news/releases/henley/pdf/henley_report.pdf

Tantau, T. (2007). *The Beamer class*. Manual for version 3.07. URL [viewed 6 Jul 2009]: http://latex-beamer.sourceforge.net/

Tversky, B., Bauer-Morrison, J. & Betrancourt, M. (2002). Animation: Can it facilitate? *International Journal for Human-Computer Interaction*, 57, 247-262. [verified 26 Jul 2009] http://www-psych.stanford.edu/~bt/diagrams/papers/tversky_betrancourt.pdf

Walsh, N. (1999). *DocBook: The definitive guide*. O'Reilly & Associates, Inc.

Michael Schlotter, Technical Manager
Centre for Power Transmission and Motion Control
Department of Mechanical Engineering
University of Bath, Bath, BA2 7AY, UK
Email: m.schlotter@bath.ac.uk