Intelligent Books: combining reactive learning exercises with extensible and adaptive content in an open-access Web application

William Billingsley and Peter Robinson,
University of Cambridge
wbillingsley@cantab.net

Abstract

“Intelligent Books” are Web-based textbooks that combine computer-supported exercises with content that is both adaptive and extensible. They impose very few restrictions on the kind of exercise that can be placed within the book, and they allow students to contribute material that they have written, and to incorporate material from the Web into the book. In this chapter, we describe the influences that affect the design of Intelligent Books. These come from looking at the roles that textbooks and course notes play in education, and economic factors that affect the sustainability of Intelligent Books – competing for the attention of users, and ensuring that network externalities do not prevent a sufficient quantity of material from being usable within the book.

Keywords

Intelligent Book, intelligent learning environment, semantic wiki.

Introduction

The book you are reading now is not an Intelligent Book. It uses the same words to say the same thing to every reader regardless of whether or not they can understand it. It cannot help readers to work through problems and it cannot say anything that is not already in the book. In 2003, the University of Cambridge and the Massachusetts Institute of Technology embarked on a joint project to develop the concept of Intelligent Books – textbooks that can model what they teach, that can gather new examples and material from users, and that can make use of existing material from the Web.

The outcomes of the project were not simply to develop a software product (although we did develop software during the project) but also to understand how economic, usability, and role issues affect the useful design of intelligent on-line learning resources. Particularly, we wanted to identify complementary features and design choices that would be able to take advantage of these factors. As we describe later in the chapter, we expect many of these design choices will come into mainstream practice through existing software gradually moving in a similar direction, rather than through our own product necessarily beating the competition.

The project set out from the beginning to develop “Intelligent Books” rather than tutoring systems, but there are reasons why we believe this is a valuable approach. At some point, any automated system for homework exercises has to be able to correct students about factual errors. This involves describing a piece of content, so it is useful if the exercise can be combined with some kind of content system. The conventional take-home resource that students use as a source of exercises and content is a textbook. So, we believed that if we were to develop intelligent on-line teaching materials, “Intelligent Books” replacing textbooks could be a more appropriate model than “intelligent tutors” replacing human tutors. This might sound like a petty distinction of terminology, but it has implications for the role the technology will fill: a tutor is usually a student's master,

whereas a textbook is only ever the student's slave.

A textbook does not send you nagging emails to do your assignments like a Courseware Management System, nor does it mark you down for requiring more assistance than another student. If a textbook were to take on those roles, the way students interact with it would probably change. As a simple example, if a textbook were to grade students for course credit, that could be an incentive for students to take their learning elsewhere and only come to the textbook when they were sure they would make no credit-losing mistakes. A textbook is also not compulsory – there is nothing to prevent a student from reading a page from a different book instead. Indeed, most students use more than just the textbook: they also use Google, Wikipedia, and many other resources. The challenge, then, is how an Intelligent Book can meet these more varied use cases, without losing the value of the textbook's traditional role.

In this chapter we describe the role, economic, and usability factors that we identified, and how these drove particular design choices for Intelligent Books. Some of the decisions that we made may appear to go against the grain of other recent research projects. For instance, Intelligent Books favour informal modelling in the content, leaving it up to individual exercises to decide what details about the student to model (and usually the exercises model the question in much more detail than they model the student) – a lot of recent research has focussed on modelling users' knowledge and skills in careful ontological detail. However, because of the open and extensible nature of Intelligent Books, there are reasons why a less restrictive model is potentially more usable and more helpful.

Background

For many university teachers, if they are going to place content and exercises online, the Courseware Management System (CMS) is where they will place them. Systems such as the commercial Blackboard and the open-source Moodle (Dougiamas & Taylor, 2003) and Sakai (Hardin, 2006) provide common tools for content to be uploaded onto the Web and to enable student interaction with the content and with each other. However, it is rarely the teacher of a course that gets to decide which CMS should be used – they are often institution-wide systems. So, an institution’s CMS faces the difficult challenge of supporting many disparate subjects across one institution, where a textbook would support similar subjects across a number of institutions. However, there has also been a growing interest in the use of non-institutional software chosen by individual teachers or even by the students themselves, thanks largely to the rise of social networking sites (Pankhurst & Marsh, 2008). This suggests there might be an opportunity for a return to the textbook model, where content and exercises can be sourced from outside the university.

There is a long history of research in technology-enhanced learning that we drew upon in our project. Almost every university engages in teaching and in computing research, and has at some point combined the two in a research project. Intelligent Books are necessarily wide-reaching applications. Even for fairly narrow courses, there are a wide variety of exercises, diagram forms, and pedagogical techniques that might be useful to include in a textbook. So, a large proportion of the previous research is useful to the design of Intelligent Books. However, this also makes it difficult to provide deep and broad enough coverage of the previous research in the short space of a single chapter’s background section. In this section we limit ourselves to introducing projects that particularly informed our work and that provide examples of techniques that we discuss later in the chapter.

Content

One of the earlier attempts to produce an adaptive Web-based textbook was ELM-ART

(Brusilovsky, Ritter, & Weber, 1996; Weber & Brusilovsky, 2001), which taught LISP programming. Lessons were derived hierarchically into sections, and a four-layered user model was kept about each section. This recorded whether a student had visited a unit, which test items they had attempted, whether the material could be inferred as being known from the user's knowledge of other units, and whether the student had manually marked the material as being understood. This information was used to make automated recommendations about what a student should study next. Peter Brusilovsky, one of the authors of ELM-ART, also wrote some early papers (Brusilovsky, 1996; Brusilovsky, 2000) investigating and categorising the methods and techniques of "Adaptive Hypermedia". For instance, whether systems alter the content within their pages or only adapt the navigation, altering the links to take users to the most relevant page for them. More recent work has included examining *social adaptive navigation* (Brusilovsky, Chavan, & Farzan, 2004) – learning from the navigation patterns of previous users.

ActiveMath (Melis *et al*, 2001; Melis & Siekman, 2004), a European project to develop an adaptive interactive textbook platform for mathematics, takes the detailed artificial intelligence based approach a step further. It keeps fine-grained semantic models of the content that it is trying to teach in a format called OMDoc (Kohlhase, 2000). As well as describing the mathematical relationships between concepts, ActiveMath's OMDoc documents also contain pedagogical metadata, for example describing how abstract a particular concept is. The system models each student against a number of competencies for each topic. By drawing on these models, ActiveMath can automatically generate personalised courses that match the learning goals and knowledge levels of individual students.

The Living Book (Baumgartner *et al*, 2004) promoted the idea of slicing an existing source, such as a set of lecture notes, to populate an adaptive learning system. An existing text is divided automatically into topic-sized slices, and the relationships between the slices are inferred from the text. The slices can be reassembled in different levels of detail, depending on the student's level of knowledge and the scenario they wish to use the book for. For example, students can examine all the exercises for a topic to revise for an exam, or can find all the references to further literature.

Questions

A very wide variety of approaches have been taken to guiding students through questions in tutoring systems. At the closest level of detail, Model Tracing Tutors, such as Andes (Gertner & VanLehn, 2000) and Carnegie Mellon University's Cognitive Tutors (Anderson *et al*, 1995), model the step-by-step process that a student should undertake in order to answer the question. Constraint Based Tutors such as SQL-Tutor (Mitrovic & Ohlsson, 1999) and CAPIT (Mayo & Mitrovic, 2001) examine the rules that students break in their answers, for example CAPIT examines the rules of punctuation. In both cases, a key output of the tutor is a fine-grained model of which steps or rules the student has understood and which they have not. At a much looser level of detail, systems such as rely on the fact that different students often make the same mistakes, so there are often common wrong answers that students will come to. Feedback for these common wrong answers can be handwritten by the teacher, and automatically delivered to the students that enter that answer.

Reactive Learning Environments were first proposed in the 1970s, with the SOPHIE system (Brown, Burton, & Bell, 1975). Students are encouraged to try out their own ideas, and receive detailed feedback based on a machine model of the scenario. In SOPHIE, students were presented with a hypothetical faulty circuit, and were able to measure circuit properties and propose theories as to what the fault might be. A machine model supplied the measurement data, and also verified whether the students' theories were consistent with the measurements they had made so far. If they were not, it could describe the discrepancy. This style of pedagogy is important because it does not assume that the modelling system itself can automatically solve the question, making it suitable for design tasks such as programming or proofs. The act of compiling a student's

code and showing a detailed error message is fundamentally a reactive approach.

Many interactive systems have been developed that can provide an educational benefit without marking or correcting the students' work at all. For example, John Billingsley's JOLLIES (Billingsley, 2001) are Web-based simulations that expose the relevant parts of the program to the student. The intention is that by altering this code and observing the results, students gain an understanding not only of the system being modelled but also of how to write simulations. The exercise does not need to examine the student actively for this learning to take place; students learn simply by seeing the results of changes they make to the program. GeoGebra (Hohenwarter & Preiner, 2007) is a geometry component designed to be embedded into Web pages as a "mathlet". It allows teachers to create dynamic geometry constructions with which students can interact. These can be made part of worksheets by surrounding the mathlet with text instructions for the students, or by scripting the component using a JavaScript API. Teachers can share their worksheets at a central community site, the GeoGebra Wiki.

Just as students can learn by assessing the results of their own work, they can also learn by comparing their work with others'. In the unpublished Exegesis system developed by CARET for the University of Cambridge, students are asked to write glosses (footnotes) for sections of Shakespearian text. Once a student has written a gloss, they can then compare it with the glosses that other students have written for the same text. To ensure that students contribute glosses, rather than only comparing others' work, they are prevented from accessing the glosses that others have submitted until they have submitted one themselves.

The Nature of an Intelligent Book

There are some characteristics of a textbook, or of the way students work with textbooks, that we believe it is important for an Intelligent Book to preserve. These are design principles that feed into the discussion later in the chapter.

An Intelligent Book is a voluntary resource.

As we described in the introduction, we see textbooks as being the servants of students. Even where teaching courses require the use of a particular textbook, or require students to answer particular exercises from the book, those requirements come from the teaching course and are not part of the textbook itself.

An Intelligent Book's exercises should be varied within subjects.

A textbook for a subject usually contains a variety of different kinds of exercise. For example, a book on digital circuits might include exercises working with circuit diagrams, timing diagrams, state charts, and simulations of how individual transistors in the circuit behave. For each of these exercises, a different modelling system might be appropriate, and it might be appropriate to use a different pedagogical technique. It would be a pity if an Intelligent Book restricted the range of techniques and questions that could be used, rather than enabling yet more possibilities.

An Intelligent Book should support multiple explanations of the same content.

This might sound like an unusual claim, as traditional textbooks often only explain material once, and indeed many notable systems such as ELM-ART and ActiveMath (described earlier) have had some success keeping a single underlying record of the material. However, in April 2007, the

booksellers WHSmith listed thirteen different textbooks for thermodynamics as being in stock, and twelve more as available on order. Most university libraries do not limit themselves to a single text on a subject, and most courses' reading lists include more than one book. Many students do not limit themselves to textbooks but also use Wikis and Web-based tutorials. So we would suggest that students do not, presently, work in an environment where there is a single canonical explanation of a piece of material, nor even an adapted explanation that is "best for them". Instead, they work in an environment where there is a competing marketplace of explanations. We believe that allowing these competing explanations to be "brought into the book" would more closely match the way students work than trying to automatically adapt a single explanation of our own and excluding all others.

An Intelligent Book should be varied between subjects.

Anecdotal experience from the textbook publishing industry is that different subjects centre around very different styles of interaction (A. Black, personal communication, 2007). For example, for physics students the centre of activity might be working through example problems, while language students find community-based interaction much more important as they need to practice speaking to each other. This suggests that if a textbook infrastructure is to be useful for different subjects, then some care needs to be taken to ensure assumptions about the way students work in one subject are not so ingrained into the architecture that it cannot be adapted for the way students work in another subject.

An Intelligent Book should be affordable.

Anecdotally, a number of computerised tutoring systems have been abandoned by the universities they were developed within simply because it was cheaper to offer additional human tutoring than to maintain and support the tutoring system. So if teaching systems are to be successful, they should consider the economic influences that will affect whether they become sustainable or not.

Some economic factors influencing the design of Intelligent Books

Content availability and network effects

A network effect, sometimes called a network externality, is where the value of a product depends upon the number of people using it (Katz & Shapiro, 1985). The classic example is the telephone: if only one person in the world owns a telephone it is useless, but as more and more people come to own telephones so each one becomes progressively more useful. As well as propelling the growth of popular products, it is also a barrier to entry for new products that have no established network.

Extensible and adaptive learning platforms can face network effects in a number of ways. For example, the first user could face a platform with no content and that has learned from no previous users. In that situation its unique selling point – the collaborative and adaptive nature of the platform – is essentially useless. So, it is important to make it easy to "seed" the platform with content at very little cost to overcome this barrier to entry. A second challenge is that platforms can find themselves competing with a larger network. For example, ActiveMath is based on high quality semantic content in the OMDoc format. Not only does OMDoc take a high level of expertise to write, but also there are also comparatively many more producers writing less formal Wiki and HTML materials as well as many more consumers reading them. So, as Claus Zinn (2006) noted, the amount of informal Wiki materials for mathematics on the Web is growing much more quickly than the content within ActiveMath. This is not to say that ActiveMath will not

succeed, but it is an economic barrier that ActiveMath is working to overcome.

Price of entry and the market for lemons

If a textbook is a voluntary resource, that is if the student can as easily choose to look material up in a different resource such as Wikipedia, then it is in competition. Just as the value of the content to the student is important, so is the outlay of effort the student must make in order to receive that content. Here, the situation is similar to what Akerlof (1970) describes in economics as the "market for lemons". In our scenario, a student finds it difficult to judge how useful a piece of content material will be to them – both whether they will understand it and whether it fits their purpose – until after they have read it. So, if they have a choice between a source that is costly to access and a source that is cheap to access, they will often choose the cheap source even if the costlier one would have been more valuable. So, cheap low quality resources theoretically may out-compete expensive high quality resources.

Particularly, this affects the use of pre-tests. These are questionnaires or miniature tests that some adaptive systems ask students to complete when they first use the system, in order to collect enough information about the student to adapt the content appropriately. However, a pre-test significantly raises the initial cost of access, in terms of the effort students must invest. If students are unable to judge in advance that the material within the book is significantly more valuable than other information on the Web, then this will tend to drive students away to other sources.

This seems to be borne out if we look at adaptive systems in e-commerce, such as the recommendation system used by Amazon.com (Linden, Smith, & York, 2003). They do not require users to enter significant quantities of data about themselves before they can first use the site, but instead use algorithms that are tailored for the fact that they have a very limited amount of information on new customers.

Spreading the technical cost

Some of the costs involved with intelligent teaching systems are technical. The software needs to be improved and maintained, as does the hardware it runs upon. An obvious way to mitigate this cost is to spread it over several courses, or across several universities. In commercial software, there has recently been an increased acceptance of "Software as a Service" (SIIA, 2000), where Web software is hosted by the vendor so that the customer does not need to have the capacity to support the software, nor the hardware capacity to run it. This is an attractive approach for Web-based education, because it allows the early cost to be spread between similar courses at different universities, rather than different courses at the same university (which would have more diverse needs in terms of the content, appropriate interaction styles, and modelling systems required).

There are convenient advantages here that the "Intelligent Book" approach has over intelligent tutoring. While universities might be reluctant to outsource their teaching or tutoring to an external provider, they already source most of their textbooks from external publishers. Also, as Intelligent Books are voluntary by nature and not used for summative assessment, some of the privacy issues of student interaction data being stored off-site are mitigated. Finally, as textbooks are usually recommended by the faculty but bought by students, externally hosted collaborative textbooks can potentially avoid lengthy university technology purchasing processes in a way that locally installed collaborative teaching software cannot.

Spreading the content cost

Traditional paper textbooks are expensive to produce. The writing and reviewing process can take

two to four years. It is unrealistic to expect textbook authors to make this kind of investment in developing material for a new technology platform until it has not only had previous deployments, but has proven itself commercially successful. So, new technologies like Intelligent Books face a bootstrapping problem that they must prove their worth cheaply before content authors will be willing to invest the cost and effort required to produce material to the same standard as published paper books.

Basic content modelling

A design goal for the Intelligent Book's content model (Billingsley & Robinson, 2005; Billingsley, 2007) was to allow each book to contain more than one explanation of the same material. Particularly, this was to allow users to contribute their own material, and so that resources and explanations that are available on the Web can be brought into the book. Effectively, we decided that rather than trying to compete with the network of low cost materials that are available on the wider Web, we wanted to make use of it. This has implications for how an Intelligent Book should model its content.

For example, it suggests that an Intelligent Book's content model should use page-level adaptation, selecting different pages to show to different users, rather than altering content within a page. It would be very difficult to alter external Web pages, and it could be confusing to users if the book had very different models for internal and external pages. So, the content model effectively becomes a Semantic Web database of metadata about pages.

It also suggests that the content model should not be a strict and detailed ontology. External pages might not fit easily into a very fine-grained ontology, and a very detailed ontology would also be a barrier to students contributing to the book – they would need to know the ontology in detail in order to fit their material within it. We were also aware that many metadata schemes for educational materials, such as LOM (IEEE, 2002), are so extensive that they are very tedious for authors to understand and complete.

The approach we took is that the content model should only expect the minimum amount of metadata that is required for the user to be able to interact with the book, but that it should accept any additional metadata that it is given. There are two fundamental pieces of metadata about a page URI that we require the user to enter when a page is added to the database:

- Students need to be able to look up content, and the machine model for a question might need to refer to content that the student appears not to have understood. So, pages require a *topic*.
- Students should be able to navigate between different kinds of content for the same topic – for example, once a student has read an *introduction*, they might wish to see an *example*, and then attempt an *exercise*. So, we also require a *type* for each page in the system. Types differ from topics in that while a book can contain an open ended number of topics, it is useful if the number of types is limited – if readers could create new types of entry at whim, then the list of entry types available for a given topic could quickly become so large that it would be unnavigable by other readers.

Some additional metadata is collected automatically. When an external page is added, the book extracts the host name from the URI so that students can navigate based on the provider of the information – *"what does Wolfram Mathworld have to say on the subject?"* Titles and excerpts are extracted from the page, and the user ID of the person who added the material is recorded. Figure 1 shows a screenshot of a user looking up a topic within the book, and indicates the actions they can take.

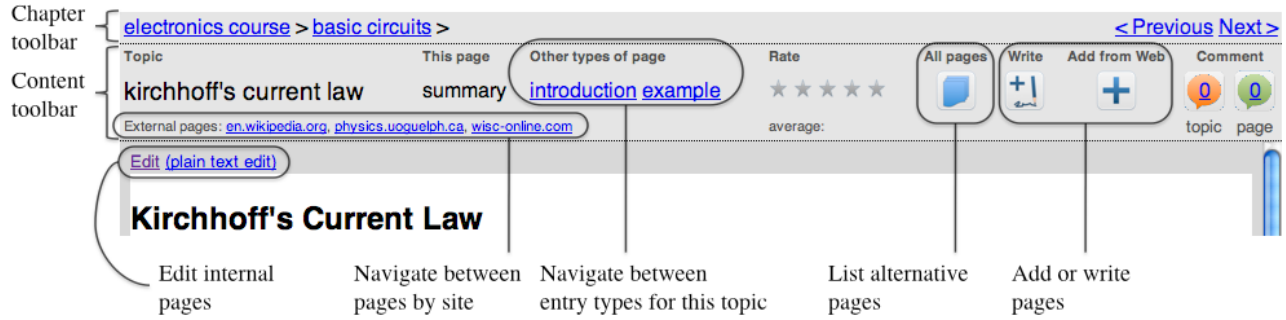

Figure 1: Two toolbars help users to work with content. The chapter toolbar navigates in an ordered way through the topics in chapters and subchapters. The content toolbar helps users to work with content for a particular topic. The user can see the topic and the type of this page, and navigation links both for the other types of page for this topic, and for pages from external sites. They can reject pages and select alternatives, write their own pages or add content they have found on the Web. They can also rate pages and comment on the topic or the individual page. Books for particular subjects might also introduce subject-specific navigation mechanisms.

Because the content model is open-ended, in that additional metadata can be added, this allows for some flexibility in the way that different books work. We said earlier that as a technology, the Intelligent Book should support variability in the way students interact in different subjects. For example, in some subjects material might not only describe a topic, but might represent a particular *view* on that topic, or might be a response to another view. For example, in economics, there might be a Keynesian view of inflation, and there might also be a monetarist response to those Keynesian arguments. It might, therefore, be important for students to be able easily to navigate between different views and responses, or even to add their own responses. Because the core metadata of the Intelligent Book is so small, and because the metadata model (represented in RDF) is flat and extensible, it takes very little code to specialise a book to support subject-specific interactions.

In addition, the user model collects data on the way each user responds to each page – from rating, to commenting, to rejecting the page and choosing another one. The selection scripts, that choose appropriate pages to show to individual students, can make use of any of this data and any metadata about the page in making their choice.

Editing structure as well as content

Textbooks are not just directories of material for particular topics. They also impose a structure and an order on those topics – the chapters, subchapters, and sections. This is what allows students to turn to page one and start reading, rather than looking up every topic from the alphabetical index at the back.

Just as it can be helpful to edit pages and add alternative pages to an Intelligent Book, it can also be helpful to be able to edit chapter structures and add alternative chapter structures – alternative routes to lead students through the material. The way we achieve this is by requiring each chapter to have a contents page, which is the first page of the chapter the student will visit. This contents page has the structure of the chapter embedded within it. This means that chapters can be stored in the content model simply as another content entry – there may be a summary of a topic and there may be a chapter listed next to it. They can also be edited in the same manner as any other internal page within the book.

When we first developed the Intelligent Book, pages were edited using Wiki mark-up. More

recently, we have moved to WYSIWYG editing, using the open source TinyMCE editor. In either case, authors of pages can easily insert links to other topics within the book. These links can be constrained on any piece of metadata – so, a link could be inserted to lookup an *example* of *Kirchhoff's Current Law*. If links in a page are marked as "subtopics", then the page becomes a chapter, and the marked links are interpreted as the structure of the chapter. When the user visits the page, and follows any of the links, the structure of the chapter will be remembered. The chapter toolbar, shown in Figure 1, appears and the user can progress through the topics using "next topic" and "previous topic" links without having to revisit the contents page. Of course one of those links might lead to another chapter, in which case the structure becomes hierarchical – subchapters and sections are simply chapters that are within other chapters. In Figure 1, “basic circuits” is a subchapter of the “electronics course”.

It is fairly easy for authors also to insert components into the page, such as diagram components for exercises, and videos from YouTube or stored within the book's content database.

Artificial separation, slicing, and narrative continuity

An Intelligent Book is a collaborative learning environment. However, the interaction between thirty students studying the same course at the same university would be quite different from the interaction between a million users sharing a textbook. Shared experiences, such as studying for the same exam at the end of the course, would be lost in a completely open textbook. There seems to be some value in artificially separating the community so that it appears as though each course has its own textbook.

We mentioned earlier that it is important to be able to prime an Intelligent Book with material quickly and easily, so that the first users are not faced with an empty book. One solution, inspired by the Living Book (Baumgartner *et al*, 2004), is to slice an existing resource such as the course's lecture notes automatically. Considering the role that lecture notes play in a course, this is very valuable practically. In a traditional course, lecture notes are not only a source of content, they also help to set expectations. From the lecturer's perspective, the course notes are designed to cover exactly the course that is being taught. Our anecdotal evidence from tutoring students is that they do look at what material is covered in the lecture notes and to what depth, in order to judge what they are expected to know on the exam. The Intelligent Book allows lecturers or other tutors to mark pages in the book as "*show first*" – that is, they will be shown to students in preference to any other page in the book on the same topic, regardless of what the adaptive selection algorithm might say. We expect that the main use for this will be lecturers or teachers who have sliced their notes and want to ensure they are shown first. They can also set a "*no additions*" flag on the topic if they wish, in order to prevent users from contributing pages at all.

Slicing the lecture notes can also help to alleviate two other issues with collaboratively written books. Firstly, there is the matter of confidence in the material. For example, a lecturer might be happy for students to be given easy access to a former student's casual but easily understood explanation of the material, but be unwilling for the system to give that casual explanation primacy over his precisely written notes. The second issue is narrative continuity. Imagine if a student was reading a topic from Site A, and when they moved on to the next topic, the book showed them content from Site B. There are likely to be some differences between those sites. Most obviously, the visual styling of the pages might look very different, and the writing styles of the authors might differ too. However, there might also be dangling threads in the narrative. For example, phrases like "continuing our example from the previous section" can be problematic if the previous section that the student saw came from a different site and used a different example.

Finally, separating the book per course allows a "rinse and repeat" step to be introduced. Rather than have each year's cohort use the same book, with forums filling up with pages upon pages of comments, a "rinsed" book can be created. This gives an opportunity for the less useful pages to be

dropped, for successful pages from other books on the same subject to be automatically transferred in, and for the lecturer to understand feedback from the way that students have been working with the book, to see whether the course itself needs restructuring. In some cases it might even be worth dropping some of the useful pages – so that the following years' cohort can also engage in the exercise of finding, discussing, and developing good explanations for the material they are learning.

Exercises

Even textbooks for narrow subjects normally contain a wide variety of different kinds of exercises. These can require working with different diagrams or notations, and need very different modelling systems at the back end – both in the way they model the subject and in the way they model the student. For some questions, it is practical to "debug the student's thinking" at a very fine level, such as is done in Model Tracing tutors. For many kinds of questions it is neither possible nor appropriate to do so. We have found that there are generally four occasions where this is the case:

1. *The steps cognitive steps to solve the question might not be known, or might be beyond the model.* For many higher order questions, and design tasks, this can be the case because the search-space of possible actions is too broad. For example, consider programming exercises: while compilers and analysis tools can check students' code, they are generally not capable of writing programs themselves.

2. *The mistakes the student makes might be difficult to relate to the model of their understanding*. An exercise we developed (Billingsley *et al*, 2004) asked students to choose appropriate circuit properties and component values for a transistor amplifier. While the student worked, a constraint propagator followed their work, using the rules of electronics to deduce what other values needed to be from the ones the student had already set. Sometimes, the model would identify an inconsistency in the student's work that involved a chain of six or more deductions, each of them on simple rules. It seems unreasonable to mark the student down on each of the six rules involved, just as it would be unreasonable to mark a student down on "understanding multiplication" because they cannot calculate 593,421 × 647,823 quickly in their head.
3. In higher-order questions, the student has not been taught the process to answer the question, but is expected to discover a way to the solution by exploring. While it might be possible to model the final solution, modelling the exploration process is an open question.

4. An Intelligent Book should be able to incorporate existing exercises that have proven effective. These might not have been designed with a detailed student model in mind, and potentially might not mark students' work at all.

We have also found that there can be a trade-off between usability and level-of-detail in exercises. For example, in earlier work, we developed two kinds of question that asked users to write proofs in number theory. One kind of question (Billingsley & Robinson, Student proof exercises, 2007) was backed by a formal model: a research-grade automated proof assistant. Although the exercises reduced the learning barrier to the use of formal proof systems, these questions were very labour intensive to develop, and there remained some usability issues. If the proof assistant gave an intermediate result (it could not verify a step but could not invalidate it either), it was hard for students to understand why. Also, some concepts about the way that proof assistant does its reasoning were difficult to communicate to students, for example what steps it would consider trivial. The second set of exercises (Billingsley & Robinson, Searching questions, 2007) asked students to construct their proofs from pre-written English language statements, hidden behind a "search and select" mechanic so that students would find it harder to guess the answer based on what statements were available. These questions were modelled using much simpler predicate logic. They were an order of magnitude quicker to develop, could support a wider range of

questions from the course, and could not give an intermediate result.

Because different teaching situations call for different kinds of exercise, the Intelligent Book attempts to make it possible to write any exercise using bespoke pedagogies, but also makes it very easy to include simple questions.

At the lowest level of complexity, support is provided for multiple choice, short answer and "massively multiple choice" (Billingsley & Robinson, Searching questions, 2007) items.

It is also easy to slightly more complex components, such as GeoGebra or JOLLIES, into editable pages. These components perform their modelling in the browser, which keeps the processing load away from the server, improving scalability. The components can read from and write to the book's student model using a simple REST API, to save the state of the students' work, and to write their findings about the student.

Some questions, however, require server-side processing – a research-grade proof assistant would be difficult to provide in JavaScript. So, the Intelligent Book also provides a detailed infrastructure (Billingsley, 2007) that allows questions to contain different diagram components, different models, and different teaching scripts. Although it requires some technical skill to write an entirely new kind of question, it is much simpler to write new instances of that question – questions that use the existing diagrams and models, but with different data, and that perhaps provide a little bit of extra analytical advice. The detailed design of this infrastructure is beyond the scope of this paper, but one general observation is worth mentioning. We observed that students often find the output of machine reasoning systems difficult to follow, as they work at a different level of detail to most students. We found (Billingsley *et al*, 2004) that specifying the diagram the student works with separately from the computer model, and automatically translating between them, was a successful approach to solving this problem. The diagram presents the mental model that the teacher wants the student to have of the problem. By pruning the output of the computer model, using the diagram as context, the machine explanation is therefore translated to the student's model of the question.

Figure 2 shows the electronics exercise we described earlier. The system is in the process of explaining a problem in the student's work that involves a chain of deductions – these are animated step by step on the diagram. If the student were to click on the rule (Kirchhoff's Current Law) in the step, that would look up the law within the book.

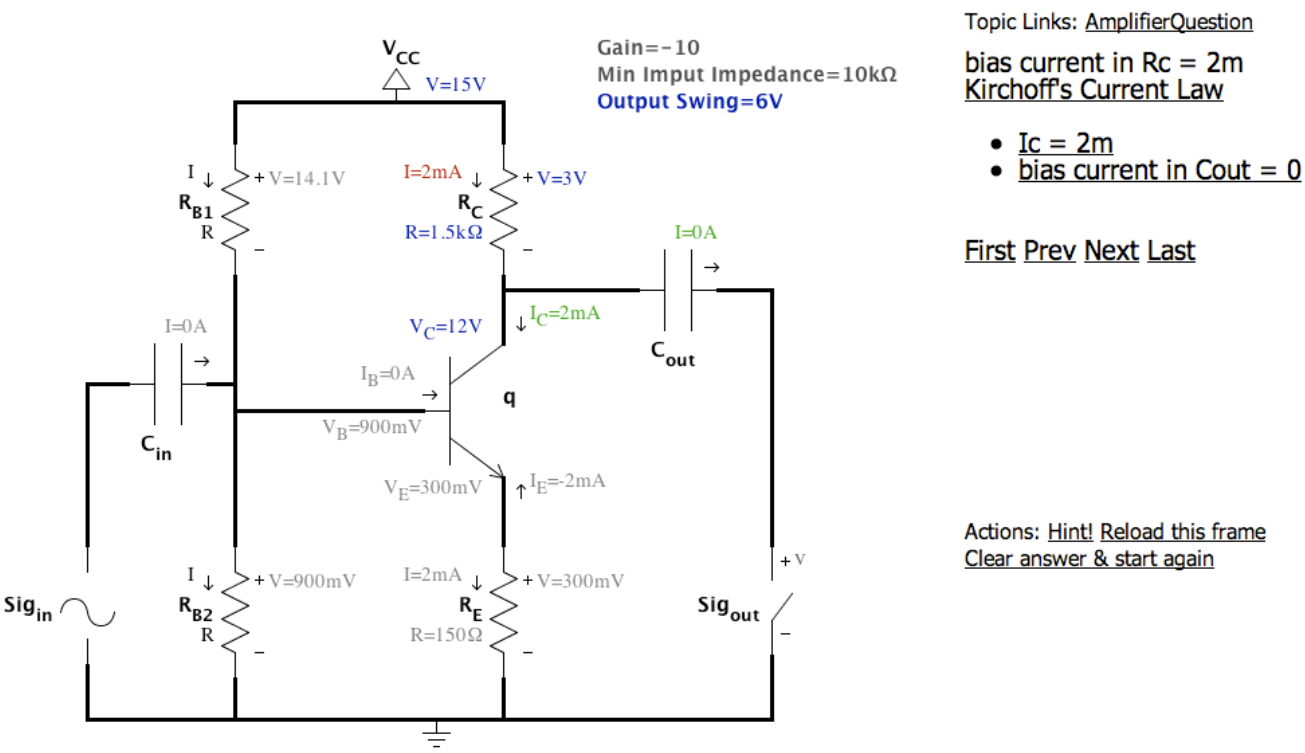

Figure 2: An electronics exercise, backed by a constraint propagation engine. The system is explaining an inconsistency it has found in the student's circuit – the steps of the reasoning engine's deductions are played back on the diagram, with each rule (Kirchhoff's Current Rule in the picture) linking automatically into the book's content.

Student Modelling and Adaptation

When we were developing the Intelligent Book, we came to the conclusion that the student modelling needed to be open-ended and *ad hoc*. Because the exercises vary so much in what they decide to model, we did not want to force a particular student modelling scheme upon them.

As we described earlier, we did not believe that it would be sensible to include a pre-test because it become an incentive to use a different resource instead – students do not need to take a pre-test to search in Google. It would also raise the effort of extending the book, as new content topics would require new pre-test questions.

We also realised that the student model would spend most of its time being out-of-date. Because the Intelligent Book is a voluntary resource, it cannot assume that it is the only teaching resource being used. We developed the system in Cambridge, where students receive regular small group human-led supervisions, in an approximation to what Bloom (1984) described as the ideal learning situation. So, far from being able to assume that students' knowledge levels remain fairly constant unless they are interacting with the Intelligent Book, we found ourselves actively hoping that they would change dramatically from use to use.

Even if we had an up-to-date student model, it would still have limited usefulness when it comes to recommending content. This is because recommending an entry is a one-shot activity. When a student first asks for a page for a particular topic, the system has the opportunity to choose the page that it believes is the most appropriate. But when the same student revisits the topic, it is important that the system shows exactly the same page again, even if new data suggests that a different page would be more useful. Otherwise, the book would become a shifting sand, constantly changing while the student's attention is elsewhere. So, recommendations have to be "sticky" in that once a

student has seen a page, it must remain in that student's book until the student explicitly asks for it to be removed. Of course, because the choice happens when the student *first* looks up the topic, it is being made just when the book knows the *least* about the student in regard to that topic. In fact, we realised that often the system will not even get to choose the page the first time. If an Intelligent Book is primed with a sliced set of lecture notes, it is quite likely that the lecturer will choose to set those lecture notes as "*show first*".

Instead, as with the content model, the book requires very little data, but collects as much as it is given. Firstly, the book records internally the interactions with each page – the rating the student gave the page, which pages they seen, which they have rejected (seen and then picked an alternative page instead), and which they have commented on. We suspect that page rejections will prove to be more useful for adaptation than page ratings – students might not rate a page, but if they do not understand one they will need to find another that they do – but we have not yet tested this theory. This collected data can then be used by the page selection scripts.

Additionally, we provide a simple REST API for recording data about students. This allows exercises that do make inferences about students' knowledge to record that data in a way that other exercises (as well as the page selection scripts) can use. However, we expect that the most beneficial use of this data will be as a reflective tool for students – what Kay (2001) refers to as scrutable student modelling. Simple reporting gadgets can be written against the student model API and embedded into "performance" pages for the topic, in a similar manner to the mash-ups that have become popular on Web 2.0 sites. These would show students how they have performed in the exercises and any areas that exercises have identified they consistently make mistakes.

Future Trends

In this chapter, we have described economic, usability, and role factors that affect the useful design of “intelligent” educational textbooks. However, there is no reason why these factors would only apply to the Intelligent Book software, where the developers have carefully examined these particular issues. Quite the opposite is true – we believe that the same factors also apply to the learning systems that universities are already using, and that over time they will push existing software in a similar direction. For instance, there is currently development work on the Sakai open source CMS to introduce a simpler way of editing content that can include adding rich “gadgets” to pages, and to add social networking tools. It is easy to imagine these efforts adding social book-marking, student-editable content, and embeddable exercises in a way that might allow students to contribute alternative explanations, link to third-party explanations of material, and use rich exercises that can refer to that content. It is also easy to imagine enterprising companies offering future versions of Sakai (or other systems) with specialised content as hosted applications – allowing university teachers to sign their classes up in the same way that they might choose a textbook for the class to use.

Intelligent Books were originally developed as a research project, but we have started to make the system available as a constantly improving prototype at *www.theintelligentbook.com*, and we would welcome approaches from teachers who wish to use the system with their classes. However, we currently do not have a marketing budget, and development occurs in conjunction with a research interest. So, we would hesitate to claim that our software would out-compete well-funded existing products. However, the design factors we have described are ones we would expect to see in educational software of the future.

Conclusion

As we have described in this chapter, an Intelligent Book needs to be quite flexible in the way that it models students, content, and exercises. Particularly it needs to support shallow and easy

modelling – so that it can take advantage of eighty-twenty rules and the large amount of simple but successful material – while being possible to add more complex modelling for specialised cases.

From looking at the ways that teachers and students use textbooks and course notes, it seems there are practical limits to how adaptive an Intelligent Book can be while still serving all the needs and expectations. Instead, Intelligent Books add value in four different ways. They allow students to work through different kinds of exercise from those that they could before, they integrate the exercises with the content in a practically useful way, they allow materials from the Web to be brought into the book, and by being collaboratively developed they reduce the cost of writing textbooks. Traditionally, writing a textbook was something only a subset of lecturers would attempt, it would take a large amount of time, and then the finished would finally be released as a fixed item for sale at a profit. Intelligent Books, however, are intended to be used before they are finished – from automatically sliced course notes, or from content from other Intelligent Books on the same topic – and are then collaboratively developed by the teachers and the readers.

We have been developing Intelligent Books since 2003. We have developed exercises from fields as diverse as number theory, electronics, and high school biology. Many of these could not be supported in more traditional or fixed ontological systems. A public prototype of the system is available, at *www.theintelligentbook.com*, and we would welcome interest from teachers and universities wishing to use it, as the proof of any flexible collaborative system is when it is used by many people, not just a few.

Acknowledgements

We gratefully acknowledge the help of our collaborators during the Intelligent Book project: Hal Abelson, Gerald Sussman, and Chris Hanson at the Massachusetts Institute of Technology, and Mark Ashdown, Kasim Rehman, and Sparsh Gupta in Cambridge. Our work was supported by the Cambridge-MIT Institute, and recent development has been carried out within the Centre for Applied Research in Educational Technology (CARET).

References

Akerlof, G. A. (1970). The market for "lemons": quality uncertainty and the market mechanism. *Quarterly Journal of Economics*, 84, 488-500.

Baumgartner, P., Furbach, U., Groß-Hardt, M., & Sinner, A. (2004). Living Book: deduction, slicing, and interaction. *Journal of Automated Reasoning*, 32(3), 259-286.

Anderson, J.R., Corbett, A.T., Koedinger, K.R., & Pelletier, R. (1995). Cognitive tutors: lessons learned. *The Journal of the Learning Sciences*, 4(2):167– 207.

Billingsley, J. (2001) Javascript Jollies Can Bring Simulations to Life, *12th Australasian Conference on Engineering Education*, Queensland University of Technology, Brisbane, Australia. pp 63-67.

Billingsley, W., Robinson, P., Ashdown, M., & Hanson, C., (2004). Intelligent tutoring and supervised problem solving in the browser. *Proceedings of the IADIS International Conference WWW/Internet 2004*, Madrid, Spain. pp 806 - 811.

Billingsley, W. & Robinson, P., (2005). Towards an intelligent online textbook for discrete mathematics. *Proceedings of the 2005 International Conference on Active Media Technology*, Takamatsu, Japan. pp 291 - 296.

Billingsley, W., & Robinson, P., (2007). Searching questions, informal modelling, and Massively

Multiple Choice. *International Conference of the Association for Learning Technology (ALT-C 2007)*, Nottingham, UK.

Billingsley, W., & Robinson, P. (2007). Student Proof Exercises Using MathsTiles and Isabelle/HOL in an Intelligent Book. *Journal of Automated Reasoning* 39, 2, 181-218.

Billingsley, W. (2007). *The Intelligent Book: technologies for intelligent and adaptive textbooks focussing on discrete mathematics*. Ph.D. Thesis, University of Cambridge.

Bloom, B. (1984). The two sigma problem: the search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13, 4-15.

Brown, J. S., Burton, R. R., & Bell, A. G. (1975). SOPHIE: A Step Towards a Reactive Learning Environment. *International Journal of Man-Machine Studies, 7*, 675-696.

Brusilovsky, P. (1996). Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 6(2-3), 87-129.

Brusilovsky, P., Ritter, S., & Weber, G. (1996). ELM-ART: an intelligent tutoring system on the World Wide Web. In: *Intelligent Tutoring Systems*, volume 1086/1996 of Lecture Notes in Computer Science, Springer-Verlag, 261–269.

Brusilovsky, P. (2000). Adaptive hypermedia: from intelligent tutoring systems to web-based education. In: *ITS2000*, LNCS, pages 1–7. Springer-Verlag.

Brusilovsky, P., Chavan, G., & Farzan, R. (2004). Social adaptive navigation support for open corpus electronic textbooks. In: *Adaptive Hypermedia 2004*, number 3137 in LNCS, 24–33.

Dougiamas, M., & Taylor, P. (2003). Moodle: Using Learning Communities to Create an Open Source Course Management System. In: D. Lassner & C. McNaught (Eds.), *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2003*, Chesapeake, VA: AACE, 171–178.

Gertner, A.S., & VanLehn, K. (2000). Andes: A coached problem solving environment for physics. In: *Intelligent Tutoring Systems*, volume 1839 of Lecture Notes in Computer Science, 133–142.

Hohenwarter, M., & Preiner, J. (2007). Dynamic Mathematics with GeoGebra. *Journal of Online Mathematics and its Applications*, 7.

Hardin, J. (2006). *The Sakai Project Final Report to the Mellon Foundation*. University of Michigan.

IEEE (2002). *Standard for Learning Object Metadata*. IEEE Standard 1484.12.1-2002.

Katz, M. L, & Shapiro, C. (1985). Network externalities, competition, and compatibility. *The American Economic Review*, 75(3), 424-440.

Kay, J. (2001). Learner control. *User Modeling and User-Adapted Interaction*. 11(1-2), 111-127.

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1), January/February, 2003, 76-80.

Mayo, M., & Mitrovic, A. (2001). Optimising ITS behavior with Bayesian networks and decision theory. *International Journal of Artificial Intelligence in Education*, 12, 124–153.

Melis, E., Andrès, E., Büdenbender, J., Frischauf, A., Goguadze, G., Libbrecht, P., Pollet, M., & Ullrich, C. (2001). ActiveMath: A generic and adaptive Web-Based Learning Environment. *International Journal of Artificial Intelligence in Education*, 12, 385-407

Melis, E., & Siekman, J. (2004). ActiveMath: an Intelligent Tutoring System for mathematics. *Seventh International Conference 'Artificial Intelligence and Soft Computing' (ICAISC)*, LNAI 3070, Springer-Verlag, 91-101.

Mitrovic, A., & Ohlsson, S. (1999). Evaluation of a constraint-based tutor for a database language. *International Journal of Artificial Intelligence in Education*, 10, 238–256.

Pankhurst, R., & Marsh, D. (2008). Communities of practice: using the open web as a collaborative learning platform. *iLearning Forum 2008*, Paris, France.

SIIA (2000). Software as a service: software on and off like a light. In: *Building the Net: Trends report 2000 – Trends shaping the digital economy*. Software and Information Industry Association.

Weber, G., & Brusilovsky, P. (2001). ELM-ART: An adaptive versatile system for Web-based instruction. *International Journal of Artificial Intelligence in Education*, 12(4), 351-384.

Zinn, C. (2006). Bootstrapping a semantic Wiki application for learning mathematics. In: Schaffert, S. & Sure, Y. (eds), *Semantic systems: from visions to applications. Proc. of the Semantics 2006 conference*, ACS, 2006.