An Interactive Self-Paced Learning Environment for the World-Wide Web

Anthony M. Sloane and Curtis Dyreson*

Abstract

Interactive, computerised self-assessment lessons can beneficially augment traditional modes of instruction and enhance learning. Our belief is that instructors will provide such lessons when they are certain that students will use them and when such lessons can be easily developed and maintained. The main contributions of this paper are to describe a tool, called `SAMaker`, that enables instructors to easily develop such lessons for delivery on the World-Wide Web (WWW), and to analyse the effectiveness of the tool.

`SAMaker` has been used in several Computer Science subjects at James Cook University to put self-assessment lessons on the WWW. The WWW is an attractive platform for lesson delivery since it can deliver lessons anywhere on the Internet, with graphics, animation, or sound, the WWW browser interface is simple to use, and many students are already familiar with and use the WWW for other reasons. `SAMaker` generates a lesson from a text file so instructors, including those who know nothing about the WWW, can create a lesson using a familiar editor (e.g., emacs) and format a lesson using a common text formatting package (e.g., LaTeX). The interactive lesson generated by `SAMaker` is a sequence of multiple-choice, true/false, exact answer, or many choice questions. Each question is a WWW *form* and can include hints or links to lecture notes. When a student answers a question, the answer is automatically checked and immediate feedback on its correctness is provided. When a question is answered incorrectly, the student can retry the question, look at the answer, obtain an explanation of the answer, or skip the question and return to it later. The self-assessment interface makes extensive use of buttons; most questions can be answered with a single mouse-click. Student answers are logged (anonymously) and instructors can subsequently analyse responses through a statistical interface generated by `SAMaker`.

Reactions to the system have been very positive. Students use the self-assessment frequently and instructors find it easy to put self-assessment on the WWW. We present a detailed analysis of a thirty-five day period, during which 360 students attempted approximately 40,000 questions.

1 Introduction

In many disciplines lecture-based instruction for groups is usefully augmented with individual work to reinforce knowledge acquisition. Computer Science makes particular use of tutorial or laboratory sessions to enable students to learn and exercise practical skills. Such sessions serve a useful purpose since instructor help is available, but it is also important for students to work on their own. In fact, some aspects of subject material may only be encountered during individual learning simply due to a lack of time in lectures and tutorials.

*Department of Computer Science, James Cook University of North Queensland , Townsville QLD 4811, AUSTRALIA, `tony,curtis@cs.jcu.edu.au`, To appear in The Proceedings of the First Australian Conference on Computer Science Education (ACSE '96), Sydney, July 1996.

To encourage individual work, a common practice in Computer Science subjects is to provide a worksheet of questions, usually chosen from a textbook. Students can then work on the questions both in and outside of tutorial. Feedback on whether a question is answered correctly is an essential component in this style of learning. Instructors typically provide answers with the question sheet, but the answers are only supplied later, long after students have attempted all the questions. This delay in providing feedback hinders learning.

In our teaching we have improved on this situation by developing interactive *self-assessment lessons* for our students. Students can work on questions whenever they like and at their own pace. When a student answers a self-assessment question, the answer is checked and immediate feedback provided. Assistance in the form of hints or pointers to lecture notes is also available. Furthermore, instructors are able to observe student performance in self-assessments so that lectures and tutorials can be more accurately aimed at problem areas.

This paper describes `SAMaker`, a tool for constructing self-assessment lessons to run on the WWW. (Similar tools for putting quizzes and surveys onto the WWW have also been developed; they will only be mentioned in passing in this paper.) `SAMaker` was designed to meet the following major goals.

- *Time and location independence*
  Students should be able to work on questions at times convenient to them rather than just at scheduled class times. Furthermore, there should not be any need for them to travel to the campus. Because `SAMaker` creates a WWW application, students can use lessons at any time and from anywhere that can connect to the WWW. A significant proportion of students have home computers and some even have full Internet connectivity so location independence is useful for many. This approach also means that distance education can be easily supported.

- *Automatic feedback and assistance*
  Time and location independence implies that instructor help will not always be available to students when they are working on lessons. Hence it is necessary for automatic feedback to be provided. Not only is it desirable for answers to be checked automatically but it should also be possible for instructors to provide hints and cross-references to lecture notes. `SAMaker` requires instructors to provide answers when they are designing lessons, and allows hints or explanatory text to be included. Any text can include links to WWW lecture notes or other materials.

- *Support for both text and graphics*
  Many subjects make extensive use of pictures or diagrams so it is important for these to be supported in addition to text. Using `SAMaker` an instructor can use the full multimedia capability of the WWW to display information so arbitrary pictures, animations, etc. are easily incorporated.

- *Low cost of adoption for instructors*
  A system of this kind will only be used by instructors if the cost of adoption is low. Instructors should be able to develop lessons using familiar editors, e.g., emacs, and text formatting packages, e.g., LaTeX. `SAMaker` is simple to use, it generates the self-assessment lesson from a single text file. Even instructors who are WWW-illiterate, can use `SAMaker` to create a WWW lesson.

- *Instructor feedback*
  Even though self-assessment lessons do not usually contribute directly to student grades, it is useful for instructors to be able to ascertain student performance on each question.

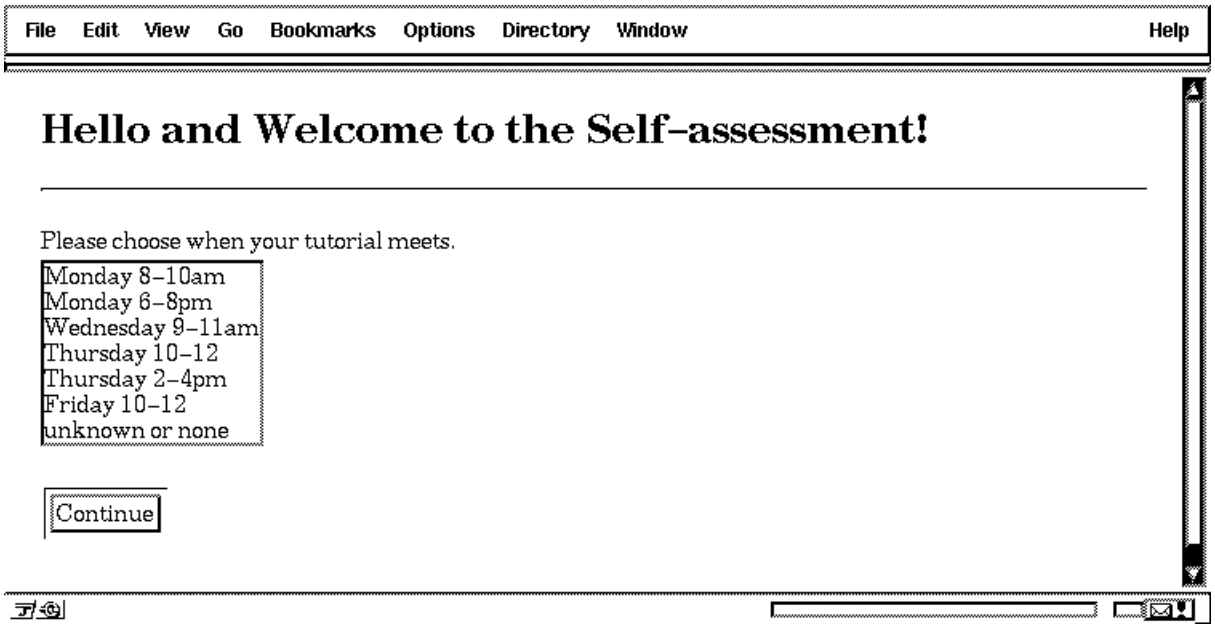

Figure 1: An example self-assessment entry page

> `SAMaker` gives instructors a complete statistical breakdown on aggregate student responses for each question. Material covered in lectures and tutorials can then be tailored based on performance in the self-assessments. Although individual student responses are collected and stored, student privacy is not violated since these responses are anonymous (students are not identified during a lesson).

The rest of this paper is structured as follows. An example use of a self-assessment lesson is presented in the next section both from the student point of view and that of the instructor. Section 3 describes how an instructor can create a self-assessment lesson using LaTeX to format the questions, and how the WWW is used to make the lesson available to students. An analysis of the approach is presented in Section 4, including usage statistics, feedback from students and instructors, and performance measurements. We then summarise related work. There are several other tools available to automatically put interactive lessons onto the WWW. Like `SAMaker` most of these packages implement "tag-based" extensions of HTML. Finally, we point to useful directions for future development.

2 A Self-Assessment Lesson

Students can access lessons produced by `SAMaker` using any WWW browser that supports *forms*, i.e., supports the HyperText Markup Language (HTML), version 1.0 or above. The screen dumps that we show in this section are from a session using a Netscape v.2.01 browser.

The lesson starts by presenting the student with a menu of tutorial groups (Figure 1). The menu choice is used to collate student responses by tutorial for later statistical analysis. No identification of the student is done other than tutorial group to minimise the connection between self-assessments and formal assessment, and to ensure student privacy.

Each self-assessment lesson consists of a sequence of questions; multiple choice, multiple answer, true/false, and exact answer questions are possible. Figure 2 shows a display for a multiple choice question used in a database subject. To answer the question the student clicks on their desired response. True/false questions are answered in a similar fashion; exact answer

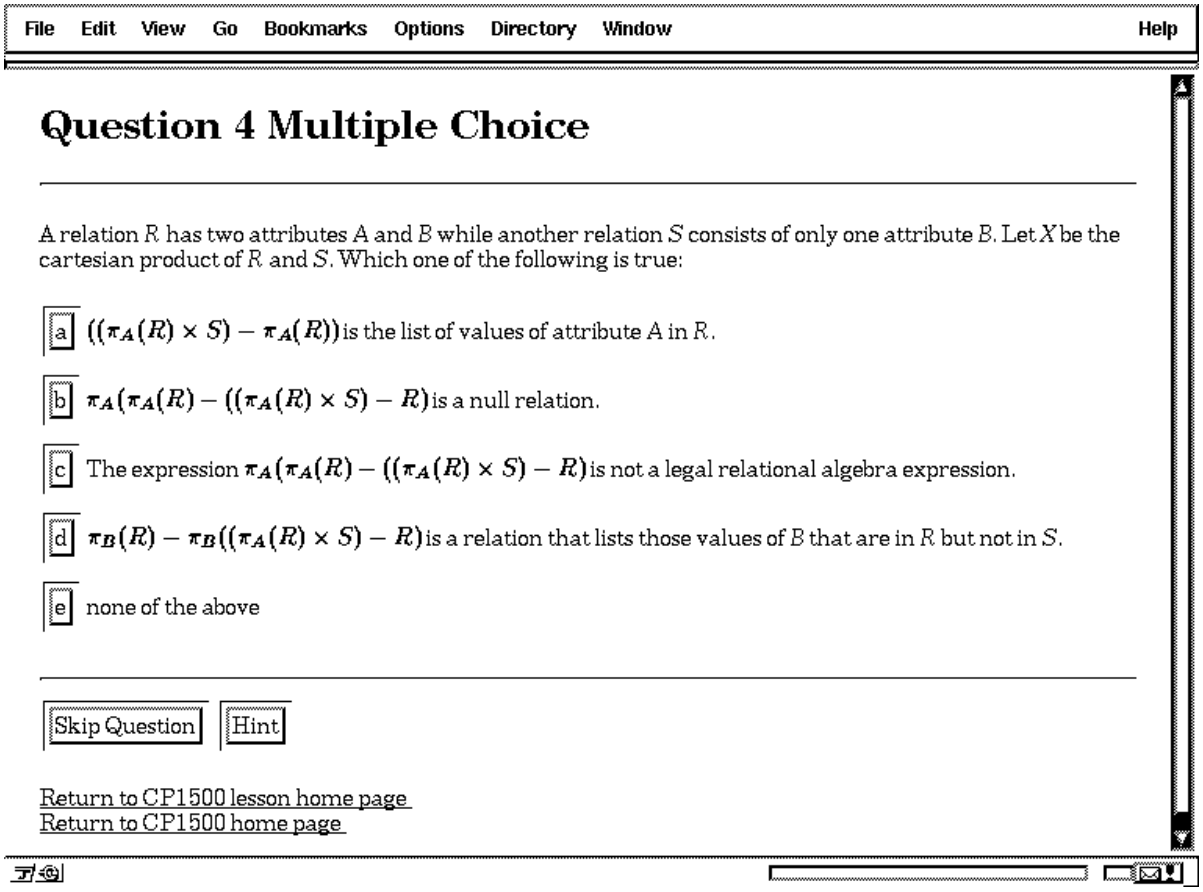

Figure 2: A multiple-choice self-assessment question

questions contain an entry area into which the answer must be typed. Links at the bottom of the question allow easy navigation back to subject pages.

Any question can be skipped and the system will automatically return to it after the other questions have been examined. If the instructor has supplied a hint for the question an extra hint button provides access to it. Any part of the question can contain WWW links to other documents such as lecture notes or auxiliary information shared by more than one question.

When a question is answered the student receives a display describing whether or not they answered it correctly. A cumulative score for this lesson is also presented. If they get a question wrong they are given the option of retrying the question, seeing what the answer is, or simply going on to the next question. In this way each student is able to tackle questions in their own way. Some students prefer to keep trying until they get it right; others like to see what the answer is and then possibly return to the question.

As answers are collected from students they are stored for later analysis by an instructor. Note that since the system does not record the identity of the student only aggregate information can be obtained. Statistics for each question is available on a per-tutorial or class basis. Figure 3 shows a display of the responses for the question in Figure 2. The total number of responses is given with an answer-by-answer breakdown of frequencies and percentages.

3 Implementation

The implementation of `SAMaker` has two main parts:

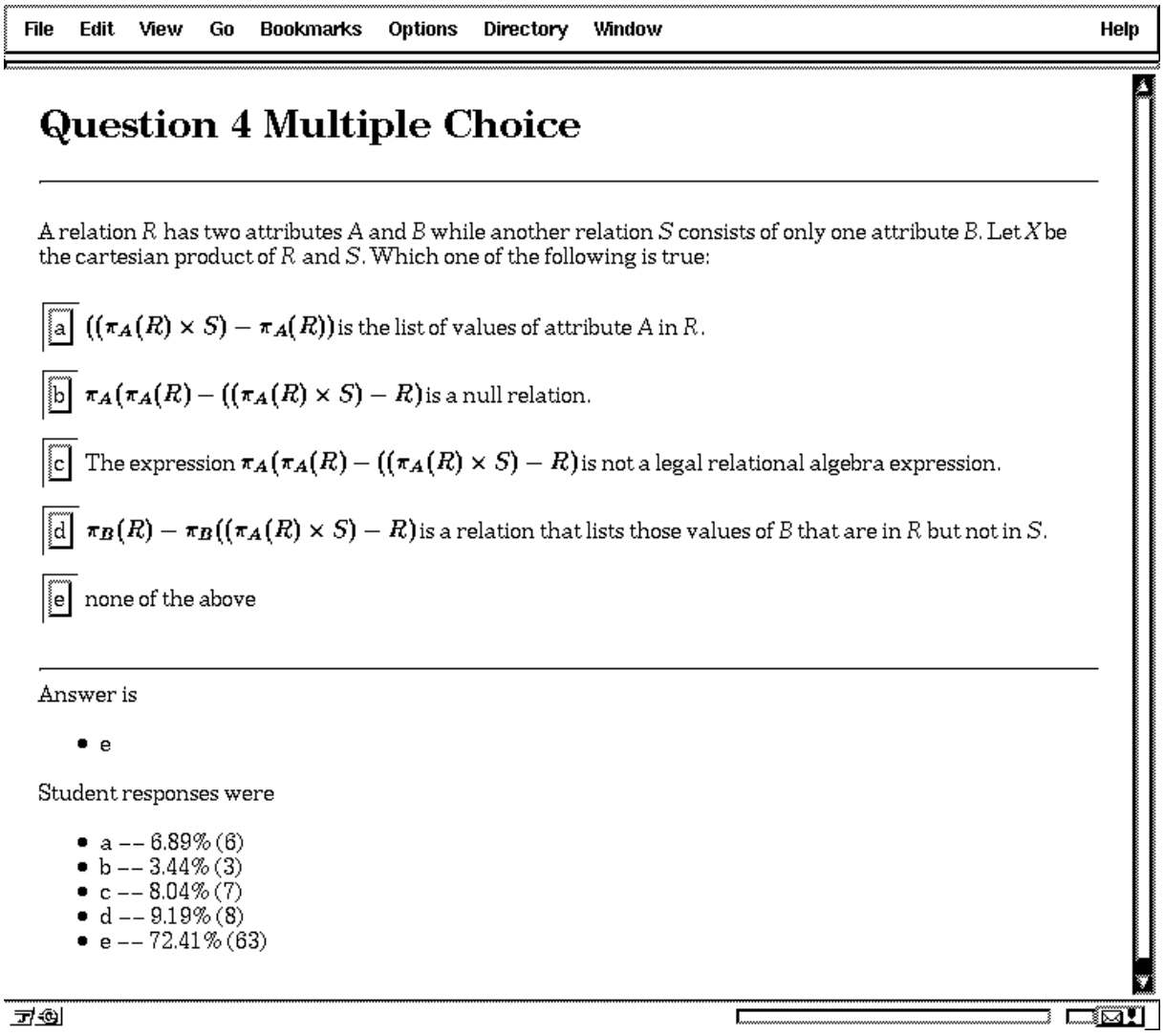

Figure 3: The statistics for a typical self-assessment question

1. a translator from input text containing questions, answers, and other markup tags to HTML described in Section 3.1, and

2. a set of scripts that are invoked by the World Wide Web to start each lesson, check and record answers, and compile statistics described in Sections 3.2 and 3.3.

3.1 `SAMaker`

Figure 4 shows the process of using `SAMaker` to put a lesson on the World-Wide Web. `SAMaker` is a Perl program that reads a text file and creates a lesson entry page, an entry page to statistics on the lesson, and several databases that contain all the information on the lesson. Command line arguments to `SAMaker` supply the name of the created pages and the directory in which the databases are created and maintained. The lesson author can delete the lesson by deleting the database directory and the created pages. Figure 4 also shows that the input to `SAMaker` can be the output of an HTML conversion engine (in this case `latex2html`). Conversion engines exist for many popular text formatting packages, so lesson authors who do not know HTML can use

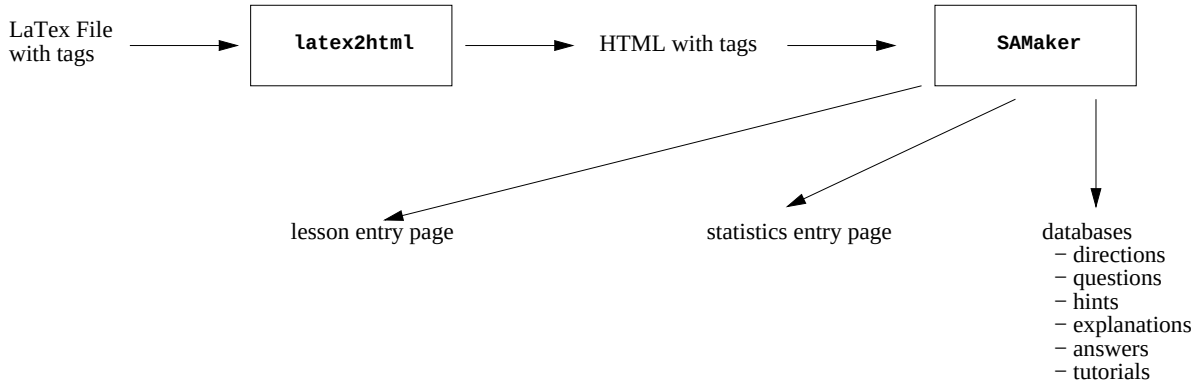

Figure 4: An overview of SAMaker

a familiar editor and text formatting package to develop a lesson.

The input to SAMaker is a text file containing both HTML and special, SAMaker tags. The HTML tags control the presentation of individual questions (e.g., to change the font of a word in a question) and pass through SAMaker unchanged. SAMaker tags, on the other hand, are interpreted and removed by SAMaker. These tags are divided into two categories: header tags and question tags. Table 1 lists all the header tags that SAMaker interprets and Table 2 lists the question tags. The header tags cover general aspects of the lesson, while the question tags are pertinent only to individual questions or groups of questions. The text associated with a tag immediately follows the tag. For example, a multiple choice question is shown below, the correct answer is indicated by the (r) tag.

```
(MC) When a tuple is deleted,
  (w) the key, if any, is replaced by NULL.
  (w) the delete operation is not allowed
      if the tuple's primary
      key is a target of a foreign key.
  (w) the tuple is deleted as well as the
      tuples that have foreign keys
      that have the deleted primary key as
      their target.
  (w) all of the above.
  (r) none of the above.
```

All header tags must appear before any question tags in the input text file. Often the same header (header tags and text) can be used for every interactive lesson in a subject.

3.2 The lesson scripts

An interactive lesson is a sequence of forms processed by two Common Gateway Interface (CGI) scripts, Starter and Marker. The first form in the sequence asks a student to select their tutorial from a menu of tutorials and is processed by the Starter script. When the tutorial selection form is processed, Starter assigns a unique identifier, which is a combination of the process id and current time, to each student and creates a statistical database for this identifier to record student activity for later statistical analysis.

Once the tutorial choice has been made, the lesson begins. All forms in a lesson are dynamically generated by Marker from the database of lesson information created by SAMaker. Each

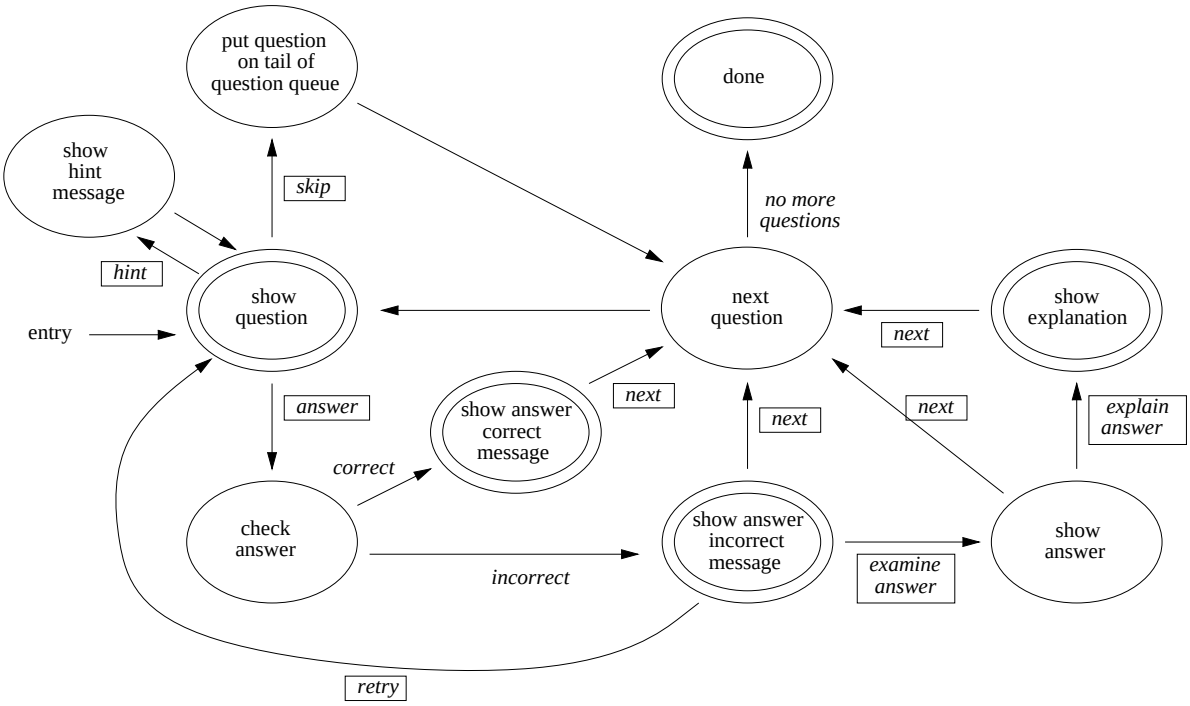

Figure 5: A state transition diagram of the marker script

question in the lesson is a separate form, with additional forms for marked questions, hints, and explanation of answers. A student progresses through the lesson by choosing available buttons on each form in the lesson. Figure 5 shows the complete state-transition diagram supported by `Marker`. States in the diagram are ovals, with terminal states denoted by concentric ovals (a student can leave the lesson from a terminal state). Transitions are represented by edges that connect states. A boxed label for an edge represents a button that is activated. For example, to move from the "show answer incorrect message" state to the "show question" state, the user would select a "retry" button. Each transition is recorded in the user's statistical profile for later analysis.

3.3 Statistics

Statistics on each user's performance in a lesson are preserved in separate statistical files. At any time, a user may obtain an aggregate view of this statistical information through forms processed by a CGI-bin script called `Stats`. Upon entry to the statistics forms, `Stats` collects all the statistical data on each question from each statistical file. `Stats` dynamically generates forms to show the percentage and number of times that a question was correctly answered and a breakdown on the variety of answers.

3.4 Modifying a lesson

We found that we would always make one or more minor mistakes when we created a lesson, but unlike a paper lesson, a WWW lesson can be dynamically modified by executing `SAMaker` again on the corrected source. `SAMaker` changes only the lesson database, and leaves all extant statistical information unchanged. Since the question number is used as a key in the statistical files, the order of questions should not be changed, although new questions can be added to the end of an existing lesson.

Tag	*Meaning*
(TUTE-LIST)	formatted list of tutorial sessions
(HTML-AT-TOP)	HTML inserted at top of every question page, e.g., to establish base directory for images produced by `latex2html`
(HTML-AT-BOTTOM)	HTML inserted at bottom of every question page, e.g., exit links
(SUBJECT)	subject code, used only to differentiate access in the HTTPD access log
(QUESTIONS)	end of header section, questions follow

Table 1: Available header tags

Tag	*Meaning*
(QUESTIONSET)	separates questions, if more than one question in set, choose one randomly
(MC)	multiple choice question
(MM)	many choice question
(EA)	exact answer question
(T)	true/false question, answer is true
(F)	true/false question, answer is false
(r)	correct answer (multiple choice, many choice and exact answer only)
(w)	incorrect answer (multiple choice and many choice only)
(h)	hint, must follow question and all answers
(e)	explanation of answer, must follow question and all answers

Table 2: Available question tags

4 Analysis

One advantage to putting a subject on the WWW is that the server access_log contains a wealth of information about whether students actually use the lessons. In this section we analyse the effectiveness of `SAMaker` by examining access_log usage of the generated lessons.

Self-assessment lessons are used in four Computer Science subjects at James Cook University: CP1020 — Introduction to Computing for Biologists (45 students), CP1200 — Introduction to Computing (144 students), CP1500 — Information Systems (103 students), and CP2001 — Data Structures (63 students).

A daily breakdown of student usage of self-assessment questions (calls to the CGI-bin scripts) for all the subjects combined is shown in Figure 6. The graph covers a thirty-five day study period, from week three through week seven during the first term of 1996. The graph plots a count of all the accesses to the department's server for each day, a count of accesses from student computers at JCU, and a count of accesses to the `SAMaker` scripts. An examination of the plot shows that students are using the self-assessment frequently and that periods of peak server usage coincide with peak periods of self-assessment usage.

Students used the self-assessments primarily to prepare for tests. Figure 7 shows a daily plot of the self-assessment accesses for each subject. Lesson use fluctuated during the study period, but is strongly correlated with tests. CP1500 accesses peak in late March, just prior to a test on April 2. Similarly, CP1200 peaks in mid April, prior to a test on April 11. CP1200 also has a periodic cycle that indicates heavy lesson use in Tuesday, Wednesday, and Thursday tutorials. CP1020 and CP2001 did not have a test during the study period so these subjects had fewer accesses.

Upon analysing these statistics, far from being concerned about a lack of student use, we became very concerned about the impact that the lessons have on server performance. Figure 8 shows a daily plot of `SAMaker` accesses as a percentage of overall accesses. On average, the lessons increased the server load by one-third, but on peak days they doubled the server load.

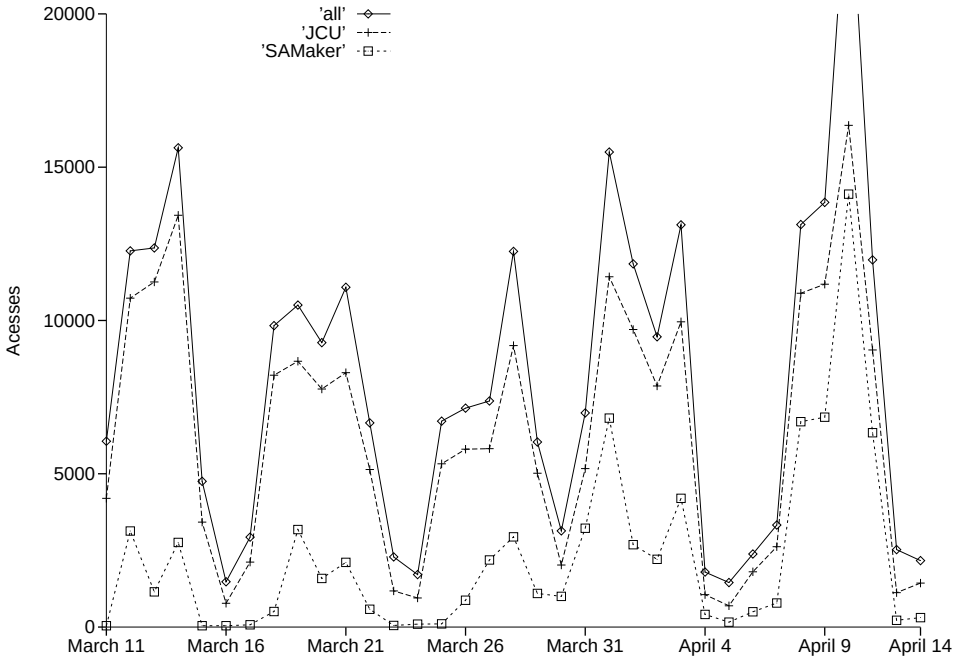

Figure 6: Daily accesses

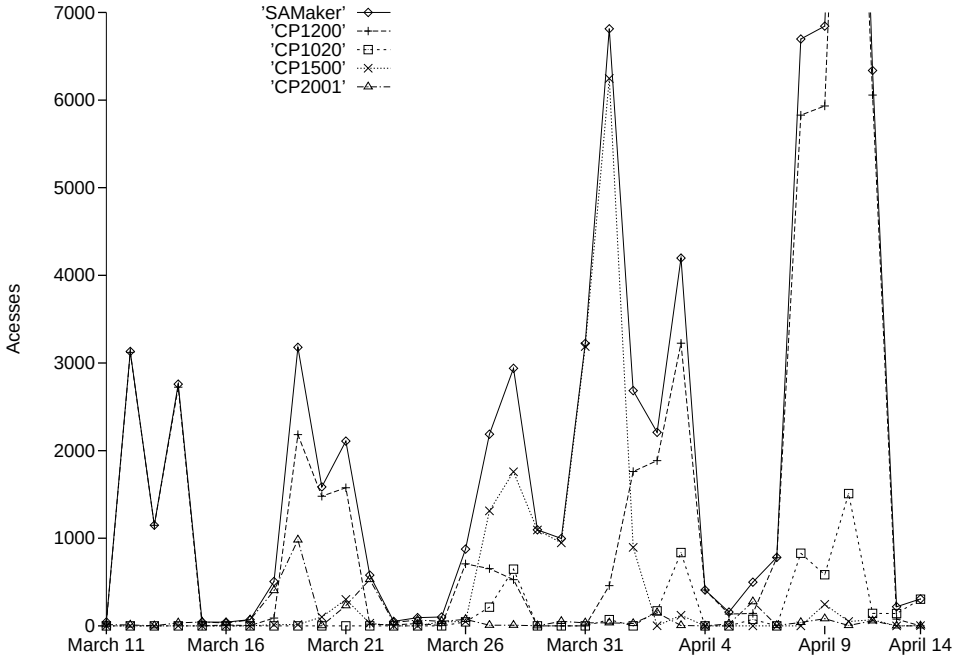

Figure 7: Daily accesses by subject

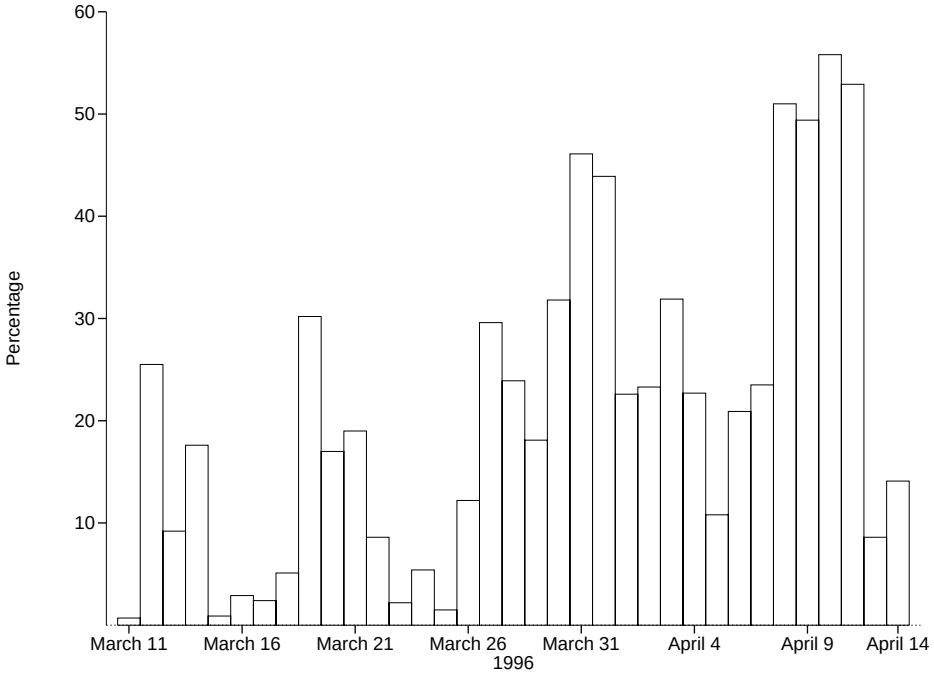

Figure 8: Daily accesses, server impact

Although the increased load has not yet resulted in a degradation of server performance (the server is a DEC-alpha), and response times remain good, the popularity of the lessons could ultimately have an adverse impact on server performance as more lessons are added.

5 Related Work

While there are many interactive lessons on the WWW there are relatively few tools to automatically convert a text file to a lesson. The tools that are available are summarised in Table 3 and described in this section.

One of the earliest and most influential lesson creation packages is the Tutorial Gateway.[1] The Tutorial Gateway is a CGI-compliant HTTPD server that extends HTML with special tags that are interpreted by the server to support interactive lessons. Unfortunately, this means that it is necessary to modify a site's server in order to use the Tutorial Gateway. Exact answer, true/false, multiple choice, and many choice questions are supported, as well as hints and specialised feedback for each possible user response in a multiple choice type question. No statistics are collected and the author must manually link a series of questions (e.g., add skip and next buttons in HTML).

The Review Automated Generation System (RAGS)[2] also uses a server modification strategy to support lessons. In addition, RAGS provides a form-based editing environment for automatic creation of multiple choice, many choice, true/false and short answer questions. Users enter questions into a central test bank using interactive WWW forms. It is difficult to enter diagrams and mathematical formulae using forms. This differs from the design philosophy behind `SAMaker`. Our philosophy is to let authors utilise existing, familiar tools (e.g., emacs and LaTeX) to develop and maintain lessons. RAGS does not support hints, explanation of answers, or statistics.

[1] *http://www.civeng.carleton.ca/ nholtz/tut/doc/doc.html*
[2] *http://stargate.jpl.nasa.gov:1084/RAGS/*

System	*Question Types*	*Presentation*	*Statistics*	*Tag-based*	*Public Domain*	*Other*
Tutorial Gateway	all except EA	(1)	no	yes	yes	(2)
RAGS	all	all at once	no	no	no	(3)
CUQuiz	MC and EA only	all at once	yes	no	yes (?)	
Mklesson	MC only	all at once	no	yes	yes	(4)
Byrnes et al.	MC only	all at once	yes(?)	yes	no (?)	(5)
Question Mark	MC only	all at once	yes	unknown	no	(6)
qform	all	all at once	no	yes	yes	
`SAMaker`	all	one-by-one	yes	yes	yes	

(1) author links questions together manually
(2) requires server modification
(3) requires server modification, maintains database of questions
(4) only non-CGI package
(5) generates test dynamically from database of questions
(6) MicroSoft Windows-based

Table 3: Summary of WWW interactive lesson packages

Like RAGS, CUQuiz[3] maintains questions in a central databank. CUQuiz provides a form-based interface to the databank that allows authors to enter and update multiple choice and exact answer questions. Questions must be formatted in HTML and typed into the form directly. CUQuiz does have a statistics package.

Mklessson[4] uses the same set of tags as the Tutorial Gateway. But instead of depending on server modification, Mklesson converts a text file containing the special tags into standard HTML. Mklesson only supports multiple-choice questions, and interprets each choice as a hypertext link. Hypertext links alone are insufficient to the task of supporting exact answer questions, statistics, and dynamic lesson navigation strategies (e.g., skip buttons).

Recently, another package that uses the same tags as the Tutorial Gateway has appeared[5], but this package uses CGI-bin scripts rather than hypertext links to navigate within the lesson. Questions in a text file with extended HTML tags are parsed and the question text, hint text, answer text, and any other information associated with a question is deposited into a centralised databank. A lesson is dynamically created by querying this databank for questions relating to a particular topic (e.g., create a lesson on 'Canada'). The created lesson is a list of links to questions, which are then answered one at a time. In common with RAGS, this package has an interactive, form-based editing module to install questions directly into the centralised databank. Unfortunately, the system is described, but no code is available. A "reporting (statistics) module" is mentioned but not described in detail.

QM Web[6] is a windows-based package for creating interactive lessons. QM Web converts lessons with multiple-choice, many choice, and exact answer questions, from a proprietary format to HTML. The lesson must be designed using Question Mark for Windows software. While Question Mark has many options, QM Web is limited. The entire lesson is converted to a

[3] Cox, K. & Clubb, O., *Formative Quizzes and the World Wide Web*, Australian Society for Computers in Learning In Tertiary Education Conference (ASCILITE), December 1995, http://ASCILITE95.unimelb.edu.au/SMTU/ASCILITE95/abstracts-/Cox.html

[4] *http://lglwww.epfl.ch/Ada/Tutorials/Lovelace/userg.html*

[5] Byrnes, R., Debreceny, R., and Gilmour, P., *The Development of a Multiple-Choice and True-False Testing Environment on the Web*, AusWeb '95, Ballina, NSW, May 1995, *http://www.scu.edu.au:80/ausweb95/papers/education3/byrnes/*

[6] Available from Question Mark Computing Ltd. at *http://www.qmark.com/qmweb.html*

single form (i.e., questions are not presented one at a time), and hints and explanatory text do not appear to be supported. QM Web has a statistics reporting package, but only the total minimum, maximum, and average mark for a lesson can be obtained.

Another lesson creation package is "qform."[7] It converts a text file with special tags to an interactive lesson form. Like QM Web, and unlike `SAMaker`, all questions in a lesson are gathered into a single form, and qform does not support hints, explanation of answers, or statistics.

Table 3 summarises the packages on which types of questions are supported, how questions are presented, whether statistics are kept for later inspection, whether special tags are used, if the code is in the public domain, and other identifying characteristics.

6 Conclusions

`SAMaker` is a successful application of World Wide Web technology to the problem of presenting class materials to students. The resulting lessons are self-paced, interactive, and are available whenever and wherever the student wants to do them, yet provide essential feedback via automatic marking, hints, explanatory text and links to online documents. Supporting self-assessment with `SAMaker` is easy because the question formats are simple and common formatting tools are supported. Collecting aggregated statistics is automatic and provides valuable information to instructors on the effectiveness of their teaching. Whenever `SAMaker` has been used, students and instructors have appreciated its flexibility and simplicity.

In future we hope to extend `SAMaker` to give the author more control over the presentation strategies, allow author defined tags, and incorporate question-specific marking modules.

Acknowledgements

We would like to acknowledge the contributions of Shyam Kapur and Marianne Brown, as well as the support of a JCU Teaching and Learning Grant for developing a WWW classroom.

[7] *http://www.satlab.hawaii.edu/space/hawaii/qform.html*