

MacQ_{TE}X: Online self-marking Quizzes, using pdf_{TE}X and exerquiz

Ross Moore

Mathematics Department, Macquarie University, Sydney

ross@maths.mq.edu.au

<http://www.maths.mq.edu.au/~ross/>

Frances Griffin

Mathematics Department, Macquarie University, Sydney

fgriffin@maths.mq.edu.au

<http://www.maths.mq.edu.au/~fgriffin/>

Abstract

The MacQ_{TE}X quiz system uses JavaScript [1, 6] embedded within PDF format [4] documents to allow students to do multiple-choice style quizzes. The internet may be used to supply the quiz document, and to record results. But even when not connected, there is immediate feedback as to how many questions were answered correctly and what are the correct answers, as well as providing worked solutions indicating how the correct answers could be deduced.

The highest quality of typesetting is employed in the quizzes by using the _{TE}X typesetting software [7], via the pdf_{TE}X variant [11], to control the generation of the PDF documents [4]. Other software, such as Perl [12] and *Mathematica* [13], can be used to control the production of unique instances of a particular quiz so that each student gets slightly different questions to answer.

— * —

PDF Quizzes

At Macquarie University the Mathematics Department has been developing¹ a web-based system for producing quizzes which allow students to test their knowledge of mathematical ideas commonly used in courses that we teach. Currently these quizzes are used mainly at the most elementary level, for revision of the basic skills which the students should have acquired from courses at high school.

The current version of this quiz facility provides students with a multiple-choice answer quiz, of typically 10–12 questions, as a PDF document [4] downloaded from a web-site (figure 1). This document is an interactive form, controlled using embedded

¹ This project has received funding via a ‘Targeted Flagship Grant’ from the Center for Flexible Learning, Macquarie University, and the Division of Information and Communication Sciences, Macquarie University as well as an equipment grant from Apple Computer, Australia Pty Ltd, via the Apple Universities Consortium.

JavaScript [1, 6], which allows a student to read and work with the document, using the Acrobat Reader plug-in [2] to his/her favourite web-browser (figure 2). Figures 2–7 show some views of such a quiz, as it appears to the student before, during and after attempting to answer the questions.

In this paper we will concentrate mainly on the _{TE}Xnical aspects of the MacQ_{TE}X quiz system. For other aspects of the full system, such as the rationale for using quizzes at all, and features available to an instructor when preparing a set of quizzes for use by students, figures 13–15 show presentation slides prepared² for talks at educational meetings.

pdf_{TE}X, exerquiz and JavaScript

A quiz document is typeset using pdf-_{La}_{TE}X [11], with the *exerquiz* [9] macros to handle the embedded JavaScript [1, 6] actions needed to produce appropriate interactivity. In this setting, JavaScript controls

- the appearance of check-boxes, as the student selects his/her answers;
- counting the number of correct choices selected, and displaying an appropriate message;
- showing which of the student’s selections were correct, which were wrong, and which were the correct choices for each question;
- resetting the form, for further attempts at the same set of questions.

Donald Story [10] has explained some of these methods elsewhere in this volume.

In fact we have added some extra features not found in the released versions of *exerquiz*. Hence the macro file that we actually use is named *exerquizX*, and *epdf_{te}x* has the coding specific to the pdf_{TE}X driver. These extra features include:

² ... using the MarSlides package [8].

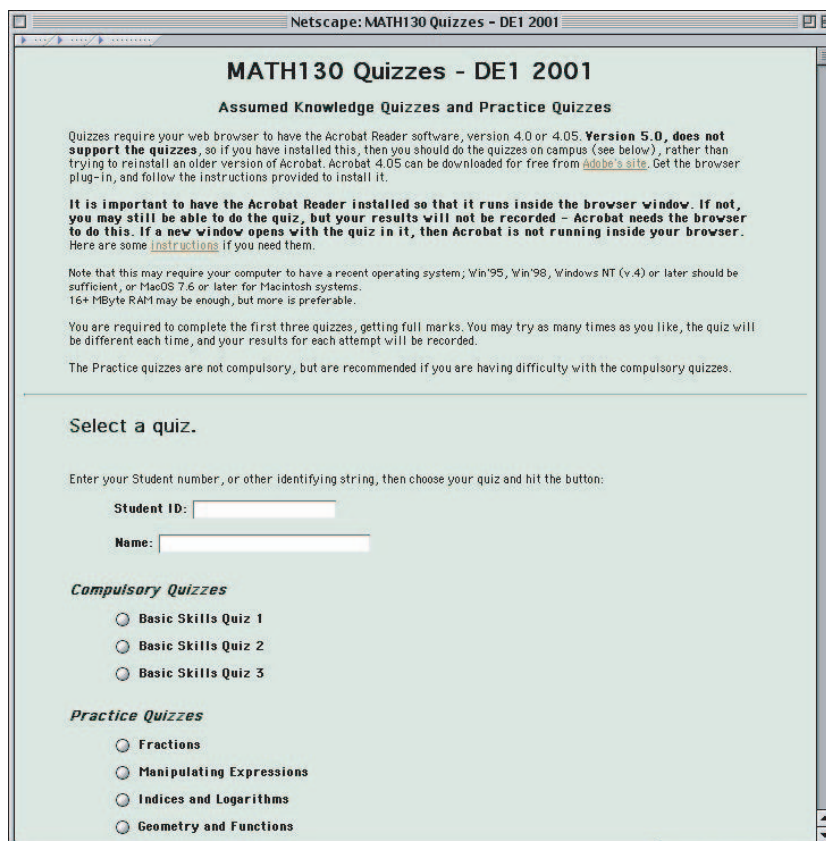


Figure 1: Quiz-site (at <http://www.maths.mq.edu.au/~fgriffin/quizzes/MATH130quizzes.html>) from which students can download the compulsory quiz documents. Username/password are required for recording accesses and results. Also from this site they may download practice quizzes, devoted to a particular mathematical concept. Guest access is also allowed for all quizzes.

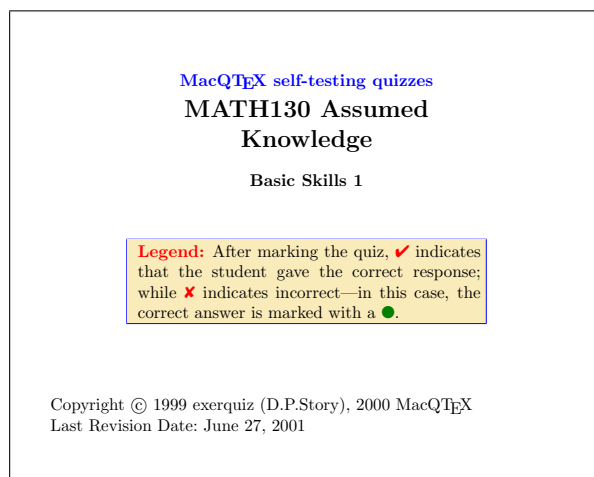


Figure 2: An opening page to a typical quiz.

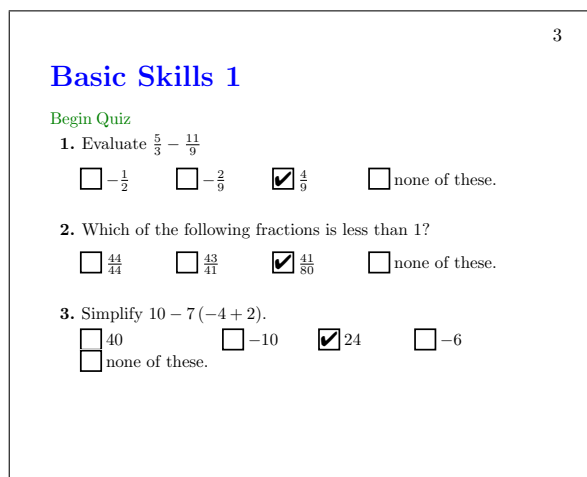


Figure 3: First page of questions, with “Begin Quiz” button and user-selections.

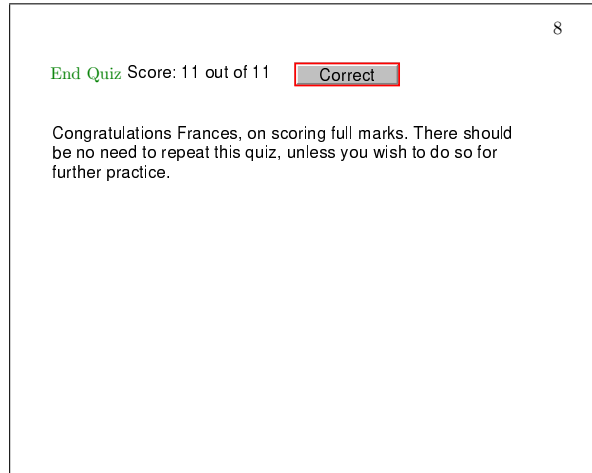


Figure 4: Last page of questions, after having selected the “End Quiz” button, showing the total score, and personalised confirmation message.

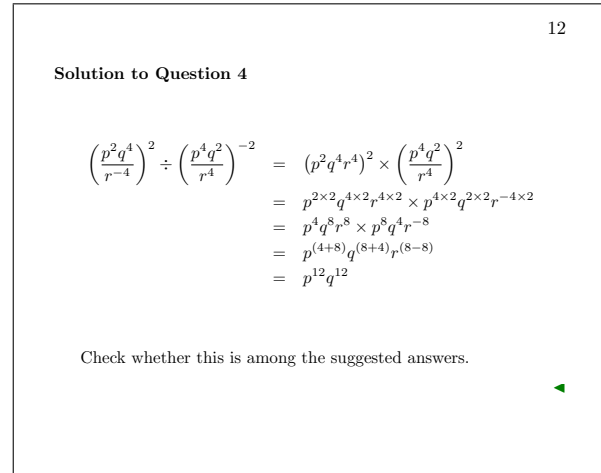


Figure 6: Worked solutions use properly typeset mathematics, as do the questions themselves. This one makes substantial use of mathematical symbols and equation alignments.

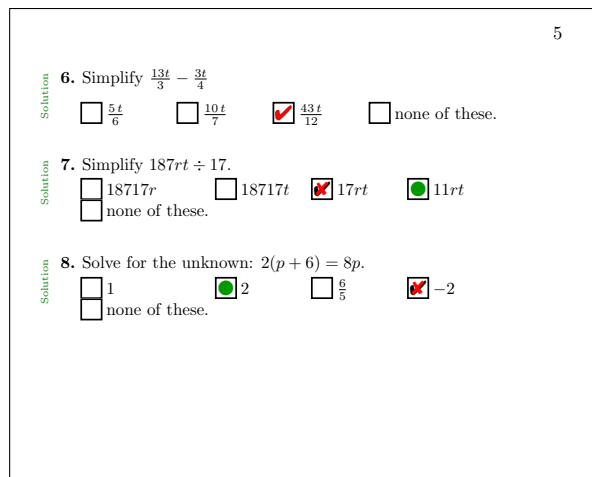


Figure 5: Embedded JavaScript [1, 6] is used to show the correct answer, when the student has made an incorrect choice. Also visible are buttons, previously hidden, which link to worked solutions.

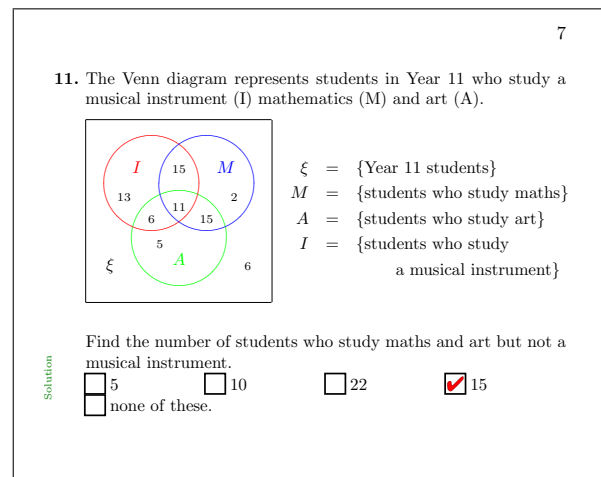


Figure 7: Elegant mathematical diagrams can also be used, both in the quiz questions and worked solutions.

- a quiz variant including both the self-marking feature *and* worked solutions;
- submission of a student’s results to a server, via the Internet;
- return of a personalised message, as feed-back to acknowledge receipt of the submitted results.

With these features, a quiz document provides a nicely typeset collection of questions and solutions which can be studied by a student, even when not connected to the Internet. The quiz can be reset for repeated attempts at answering the questions.

Only the first attempt can be recorded, and even then only if there is an active network connection.

Concerning the worked solutions, these must remain inaccessible until the quiz has been completed and the results submitted. To do this, the worked solutions are typeset in a separate document, then each is imported as a separate image to be the icon for a button field. This button can be shown or hidden under the control of embedded JavaScript code. As well as allowing the solutions to be viewed, a hyperlink from the question to its solution becomes accessible. This is done by hiding an opaque button

which otherwise screens the active area of the link from receiving mouse-clicks.

Another novel aspect of the MacQ_{TEX} quiz system is the use of randomness to produce a slightly different quiz for each student access. Currently this is done using the *Mathematica*³ [13] software, programmed to write a file of T_{EX} definitions for each question. Other ways of doing this could be used; e.g. other software could be used, or sets of \definitions could be read from a file which has been generated in advance specially for this purpose.

Designing a new quiz

Constructing a new quiz is done at a MacQ_{TEX} quiz-site, using an HTML form, as shown in figure 8. Here an instructor may choose from pre-prepared question topics; currently there are 14 such types available, covering areas of basic mathematics.

For each topic, there are up to 6 actual question types. Both the number of topics and question types for a topic can be easily extended, though some knowledge of T_{EX} or L^AT_{EX} is needed to do this. When randomisation is required, then some knowledge of programming in *Mathematica* is also useful. For a successful quiz, it is necessary to generate plausible incorrect answers, as well as the correct answer. Generating these in *Mathematica* can be an interesting challenge. Sometimes it is necessary to discard random choices where an answer intended to be wrong actually agrees with the correct answer; that is, obtaining the correct answer by a completely fallacious method.

Figure 9 shows the work-flow for making a new quiz. This includes loops for producing example quizzes, and for editing of L^AT_{EX} sources for word-ing and/or layout. Only when the instructor is completely satisfied should a batch (e.g. 50) of quizzes be generated, and made available for student access.

L^AT_{EX} source code

Figure 10 shows the directory structure at a quiz site. It can be seen that there are many files with .tex suffix, which need to be read as part of a type-setting job. Reading all of these files in the correct order is essentially a bootstrap process, in which \definitions are made as required, before the next file is \input.

For example, the main job for the quiz which was used for figures 2 to 6 uses a file having the name MATH130quiz1.tex, as follows:

```
\def\recipient{}
```

³ *Mathematica* is a trademark of Wolfram Research Limited.[13]

```
\def\defsdire{newquiz50/}
\def\texdir{../}
\nonstopmode
\catcode'\@=11
\edef\eq@author{\recipient}
\edef\eq@keywords{version 50}
\catcode'\@=12
```

```
\def\loginID{version 50}
\def\whichquiz{MATH130quiz1}
\input \texdir user.tex
```

The number 50 that occurs here is because 50 instances of this quiz have been generated. For each instance the number would have been different.

The file user.tex is constant, for all quiz instances:

```
\def\author{Fran}
\def\imagedir{\texdir}
\input \texdir a.tex
\def\quizname{Basic Skills 1}
\input \texdir b.tex
\input \texdir bb.tex
\input \texdir c.tex
\input \texdir z.tex
```

Those files a.tex, b.tex, bb.tex and z.tex are constant for all instances of a particular quiz. Indeed b.tex and c.tex are created by a Perl [12] script, and hard-code variables such as the title of the quiz and its topic. It is a.tex which contains the \documentclass command, and loads the (modified) exerquiz package, as well as other standard L^AT_{EX} packages. Similarly, the \end{document} is in z.tex. These two files are simply copied from a global storage location.

Information for the opening page of a quiz is contained in b.tex. This file could well be edited to alter the instructions, or to convey other information about the quiz. The main information in bb.tex is the URL to which results submissions should be sent. Other specialised T_{EX} definitions can be added here, when not suitable to be included in other packages. It is thus c.tex which controls input of the questions themselves, as follows:

```
% You may edit this file to change the order of
% the questions, or adjust spacing and pagination.

\def\answerdir{\defsdire}
\def\CheckIfGiven
{Check whether this is among the suggested answers.}
\def\bfr{\mathbf{R}}
\def\bfZ{\mathbf{Z}}
\def\errOnSend
{Your quiz results have not been received.}
%
\begin{quiz}*\{\whichquiz}
%
\begin{questions}
```

Staff Login

Login name:

Choose a task

- ☒ Make a new quiz
- ☐ Edit an existing quiz
- ☐ Recreate a used quiz
- ☐ Create login page
- ☐ View student logs
- ☐ Clean up directories

Look at these

[Question catalogue](#)

[Sample student login](#)

Quiz title, eg MATH130 Assumed Knowledge:

Quiz topic, eg Basic Skills 1:

Name on disk, eg MATH130quiz1 (no spaces or punctuation please):

Choose the questions

☒ A standard quiz

	Topic	Example
Question 1	<input type="text" value="absolute value"/>	<input type="text" value="1"/>
Question 2	<input type="text" value="equation solving"/>	<input type="text" value="3"/>
Question 3	<input type="text" value="factorization"/>	<input type="text" value="2"/>
Question 4	<input type="text" value="fractions"/>	<input type="text" value="4"/>
Question 5	<input type="text" value="functions"/>	<input type="text" value="1"/>
Question 6	<input type="text" value="geometry"/>	<input type="text" value="1"/>
Question 7	<input type="text" value="indices"/>	<input type="text" value="1"/>
Question 8	<input type="text" value="inequalities"/>	<input type="text" value="1"/>
Question 9	<input type="text" value="lines"/>	<input type="text" value="3"/>
Question 10	<input type="text" value="logs"/>	<input type="text" value="1"/>

Figure 8: This web-page is used by instructors and course coordinators to design quizzes for the students to use. Questions can be chosen from pre-defined categories; each category has up to 6 choices of question.

```

\item \input{\texdir question1.tex}
\medskip
\vfil
\goodbreak
\vfilneg
\item \input{\texdir question2.tex}
\medskip
...
...
\goodbreak
\vfilneg
\item \input{\texdir question11.tex}
\medskip
\vfil
\goodbreak
\vfilneg
\end{questions}
\end{quiz}
%
%
\TextField[name=\whichquiz,width=1.25in,
  align=0,bordercolor={1 1 1},
  default=Score:,readonly=true]{}%
\raisebox{3.5pt}{\quad\eqButton{\whichquiz}}
\therearequizsolutionstrue

```

```

\medskip
\TextField[name=progress,height=3cm,width=10cm,
  align=0,bordercolor={1 1 1},default=\errOnSend,
  multiline=true,readonly=true]{%

```

From this it can be seen that the source code for the questions themselves is contained in files named `question1.tex`, ..., `question11.tex`. This also contains the L^AT_EX source for the worked solution. One of these looks like:

```

\def\setupvalues{%
\def\fracsuma{1}\def\fracsumb{2}\def\fracsumc{5}
\def\fracsumd{9}\def\fracsump{9}\def\fracsumq{2}
\def\fracsumr{18}\def\fracsums{19}
\def\fracsumsign{+}
\def\ok{\frac{\fracsums}{\fracsumr}}
\def\wronga{3}\def\wrongb{\frac{6}{11}}

\IfFileExists{\defsdirefs\thequestionno}
{\input{\defsdirefs\thequestionno}}{\setupvalues}

Evaluate
${\frac{\fracsuma}{\fracsumb}}
\fracsumsign {\frac{\fracsumc}{\fracsumd}}$

```


[illegible]

In the above \TeX coding, the role of `\setupvalues` is to provide default definitions, just in case there is no file `defs\thequestionno` (that is `defs1`, `defs2`, ... according to the question number) within the `\defsdire` directory on the local file-system. If it does exist, (e.g., having been written by *Mathematica*⁴, after performing calculations employing a random-number seed) then it should contain the necessary `\definitions`. Similarly there can be a file `\answerdire answers\thequestionno` to govern the order in which the answers are presented in the quiz question.

References

- [1] Adobe Systems Inc.; “Acrobat Forms JavaScript Object Specification, Version 4.0”; Technical Note #5186; Revised: January 27, 1999.
- [2] Adobe Systems Inc.; Acrobat Reader, viewer for PDF format [4] documents, available free of charge from <http://www.adobe.com/>.

⁴ *Mathematica* is a trademark of Wolfram Research Limited.[13]

- [3] Adobe Systems Inc.; “FDF Toolkit Overview”; Technical Note #5194; Revised: August 10, 1999.
- [4] Adobe Systems Inc.; “Portable Document Format, Reference Manual, Version 1.3”; March 11, 1999.
- [5] Adobe Systems Inc.; “pdfmark Reference Manual”; Technical Note #5150; Adobe Developer Relations; Revised: March 4, 1999.
- [6] Netscape Communications Corporation; Netscape JavaScript Reference, 1997; online at: <http://developer.netscape.com/docs/manuals/communicator/jsref/toc.htm>.
- [7] Lamport, Leslie; L^AT_EX, a Document Preparation System. This is free software available for all computing platforms. Consult the T_EX Users Group (TUG) website, at <http://www.tug.org/>.
- [8] McKay, Wendy and Moore, Ross; “PDF presentations using the Marslide package.” T_EX Users Group 2001 Proceedings, (elsewhere in this volume).
- [9] Story, Donald; exerquiz & AcroT_EX, packages for including special effects in PDF documents, using T_EX and L^AT_EX. Dept. of Mathematics and Computer Science, University of Akron. Software available online from <http://www.math.uakron.edu/~dpstory/webeq.html>.
- [10] Story, Donald; “Techniques of Introducing Document-level JavaScript into a PDF file from a L^AT_EX source.” T_EX Users Group 2001 Proceedings, (elsewhere in this volume).
- [11] Thành, Hàn Thê; pdfT_EX, free software for generating documents in PDF format, based on the T_EX typesetting system. Available for all computing platforms; see <http://www.tug.org/applications/pdftex/>.
- [12] Wall, Larry; Perl, a general purpose scripting language for all computing platforms. This is Free Software, available from <http://www.perl.com/>.
- [13] Wolfram Research Inc; Mathematica, a system for doing Mathematics by computer. Consult the website at <http://www.wri.com/>.

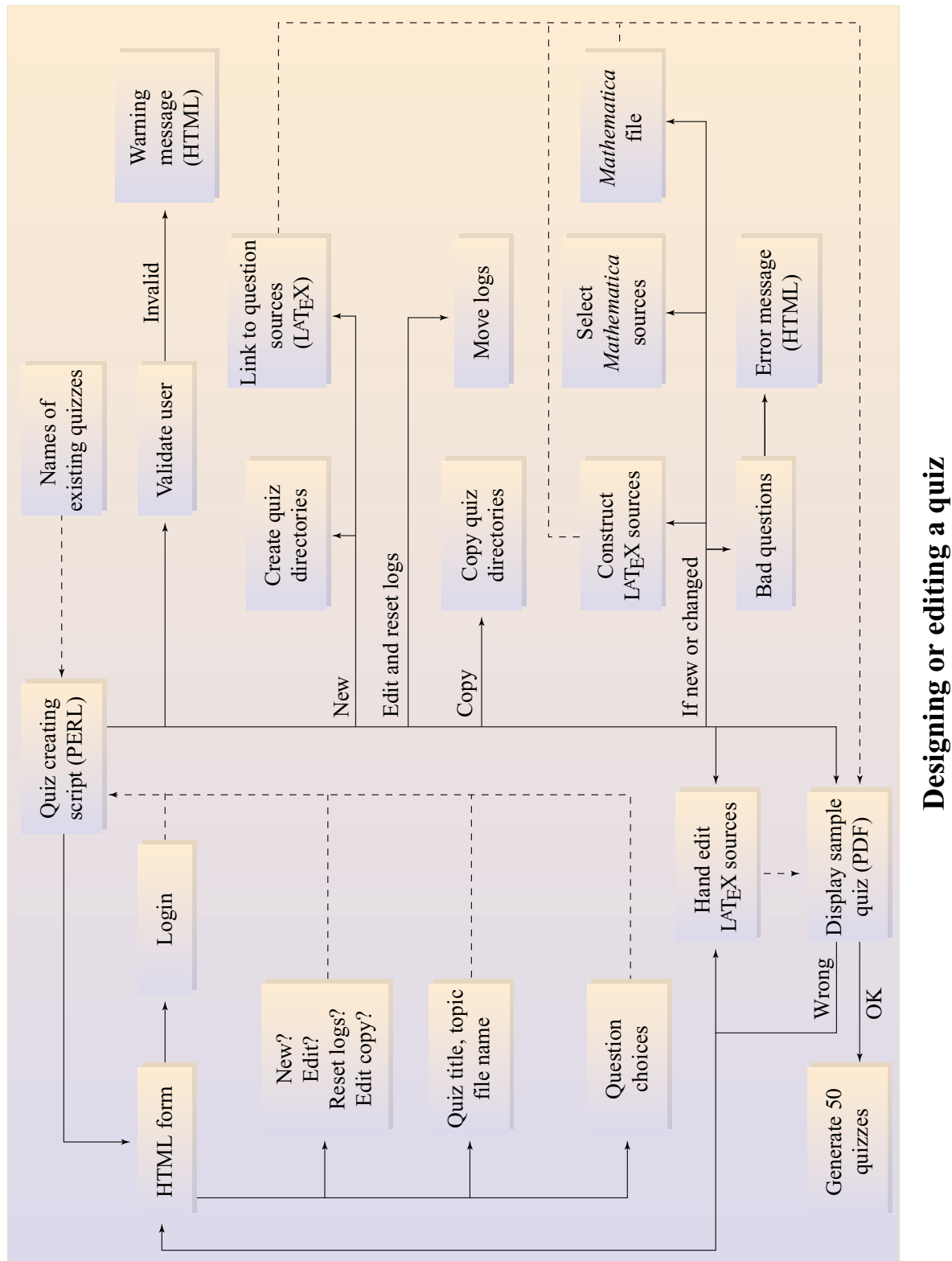
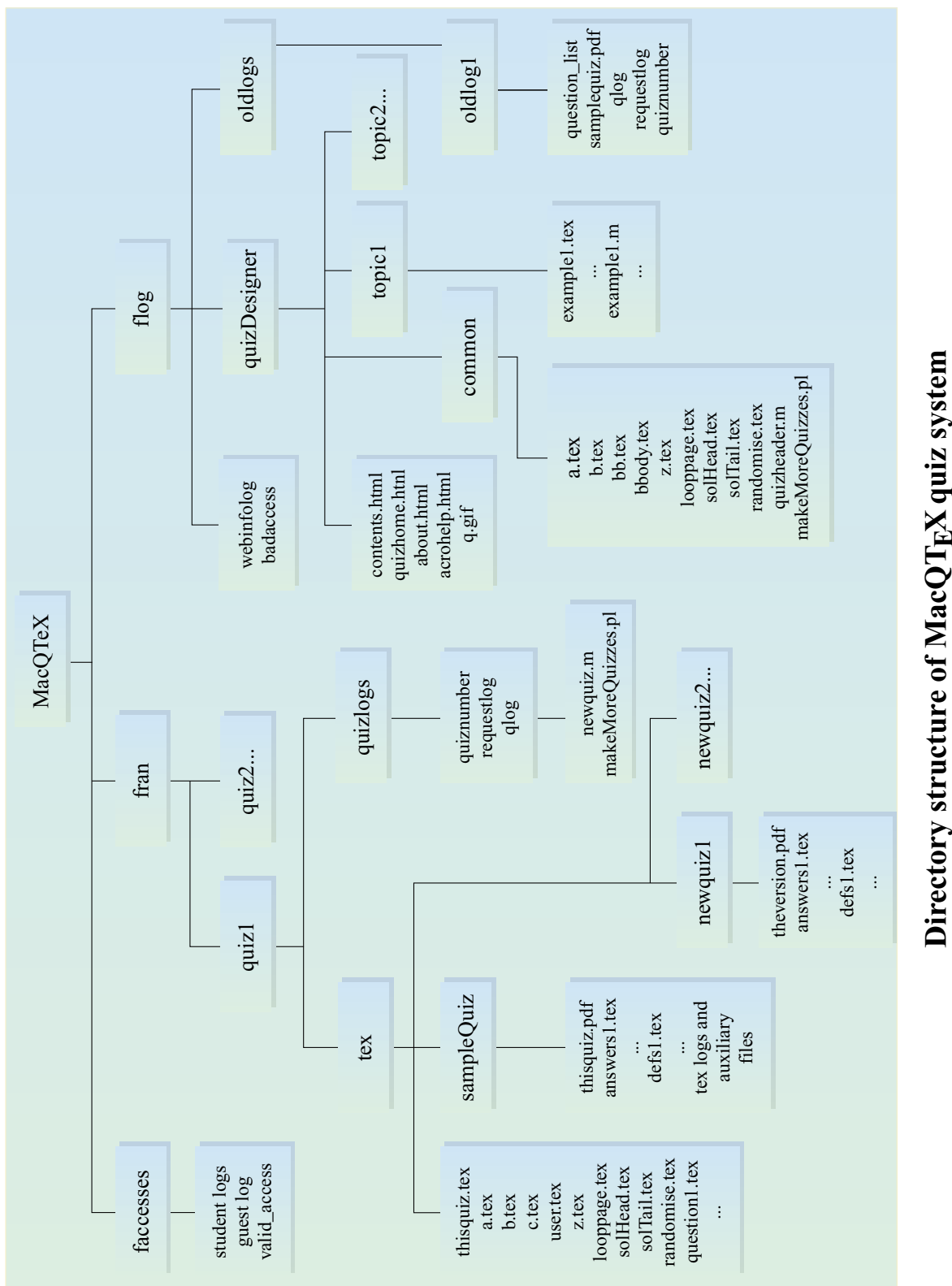


Figure 9: Work-flow for designing a new quiz, based on selections made from the HTML form shown in figure 8. Copies of existing files can be edited to taste. Constructing new question types requires more substantial editing, especially when randomisation is required.



Directory structure of MacQTeX quiz system

Figure 10: Directory structure of a quiz site, showing the locations of all the (L^A)T_EX input files for each question and quiz instance.

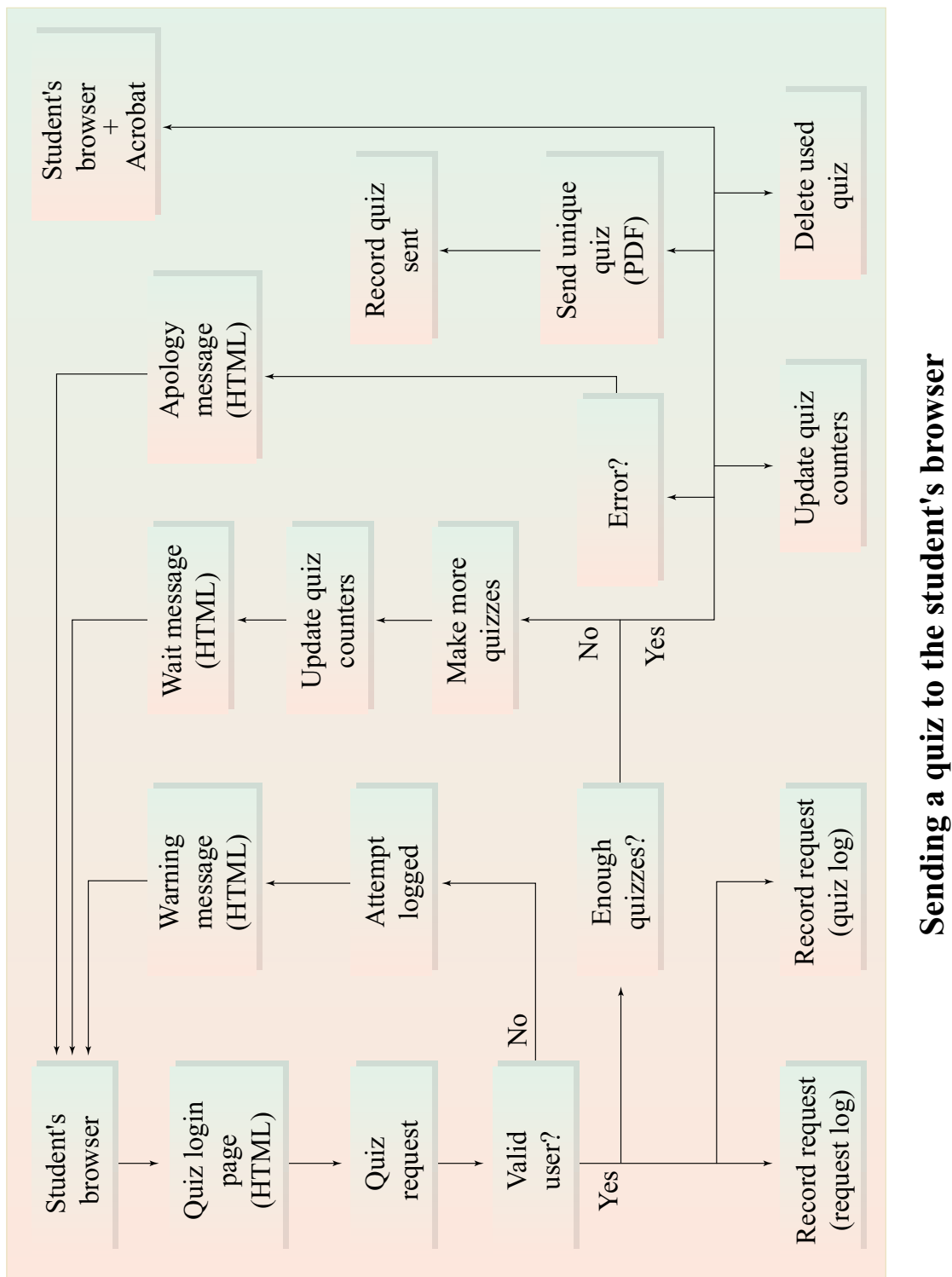
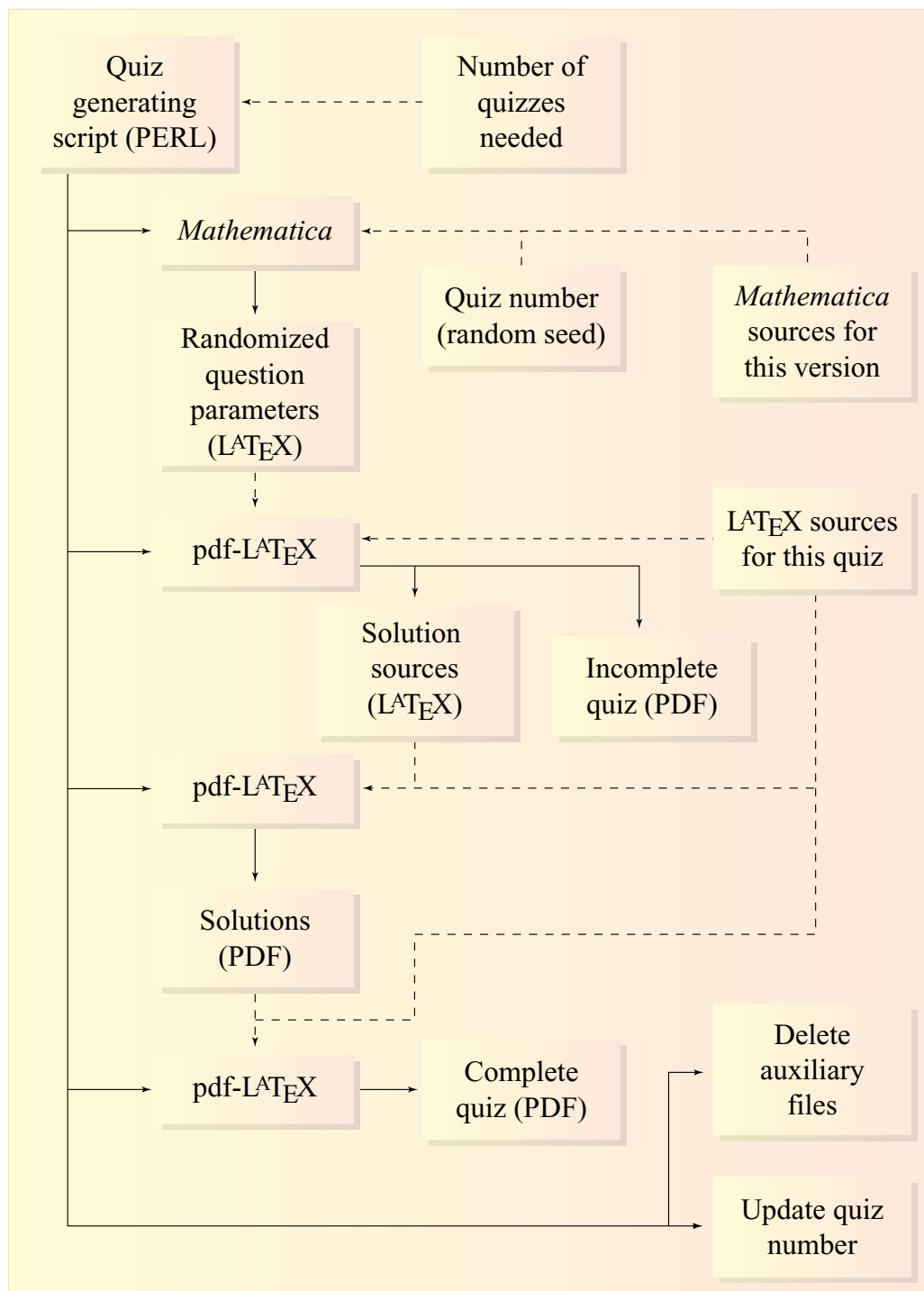


Figure 11: Detailed work-flow for sending a quiz to the student's browser. Only at the step "make more quizzes" does pdf-L^AT_EX come into play within this work-flow, as otherwise sufficiently many quiz documents have been typeset, awaiting requests for downloads. See figure 12 for the work-flow when it is necessary to generate quiz documents.



Generating new quizzes

Figure 12: Details of the sequence of calls to pdf-L^AT_EX for generating a quiz instance, along with its worked solutions. The first call causes a new document to be made containing the L^AT_EX coding for worked solutions. These are typeset at the 2nd call. Finally the 3rd call constructs a PDF document containing both the quiz questions and each worked solution, imported as a single graphic.



Randomized Quiz System

Ross Moore & Frances Griffin
Mathematics Department
Macquarie University, Sydney




Mathematics Quizzes

At Macquarie University the Mathematics Department has been developing¹ a web-based system for producing quizzes which allow students to test their knowledge of mathematical ideas required in the courses that we teach. Currently these quizzes are used mainly at the most elementary level, for revision of the basic skills which the students should have acquired from mathematics courses at high school.

Examples

- Basic Math Skills: [MATH130](#), [examples](#)
- Discrete Math Quizzes, with answers: [MATH237](#)

¹This project has received funding via a 'Targeted Flagship Grant' from the Center for Flexible Learning, Macquarie University, and the Division of Information and Communication Sciences, Macquarie University as well as an equipment grant from Apple Computer, Australia Pty Ltd, via the Apple Universities Consortium.

Training not Testing

The aim of the quizzes is not so much for assessment as for self-testing and practice of material covered previously in lectures or back at school. Why use quizzes?

For various practical reasons, related to the students ...

- wide range in ability and mathematical background;
- not their main area of study, so need to identify holes in their mathematical knowledge;
- several years since last studied any mathematics;
- maybe only a refresher course may be needed;
- insufficient staff to help every student.

The quizzes can be used by students to identify for themselves where they are weak and may need to seek the extra help that *can* be provided.

Student Interaction



```

graph TD
    Request([Request]) --> Student[Student]
    Student --> Quiz[Quiz]
    Quiz --> Validation([Validation])
    Validation --> Request
    Student --> Result[Result]
    Result --> Delete([Delete quiz])
    Result --> Logs1([Logs])
    Student --> Progress([Progress message])
    Progress --> Submit([Submitting quiz result])
    Submit --> Logs2([Logs])
  
```

When a student requests a quiz, the identity is first validated; if authorized, a quiz is sent to the student's browser. A record is kept of all request details.

After completing the quiz, submitted results are recorded in the student's log and in the overall quiz log. A personalized message concerning the student's progress is returned as FDF data. This appears in a form field at the end of the PDF quiz document.

Figure 13: Presentation slides, mainly for use at education meetings. These give the rationale for using quizzes, as well as giving a rough representation of a student's interaction with the system. Also one slide has active hyperlinks to some actual web pages from where quizzes can be downloaded.



Figure 14: These presentation slides indicate how a random aspect is incorporated into the quizzes. Also shown are other aspects of generating quizzes, obtaining statistical information from the log-files, and general house-keeping duties that can be performed.

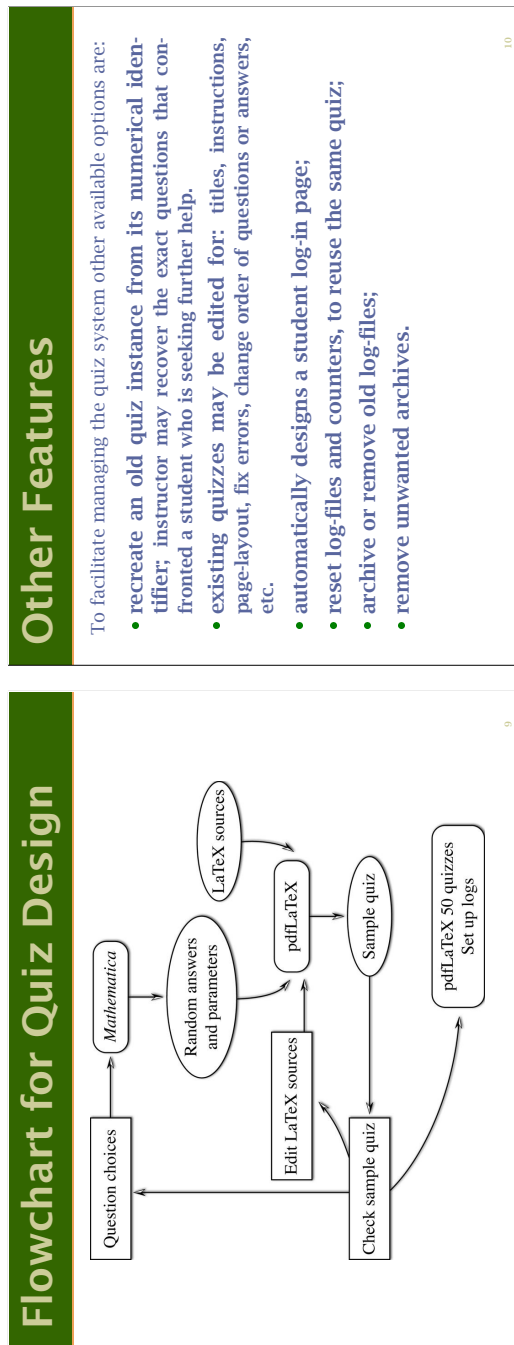


Figure 15: One slide here gives a rough view of the work-flow involved when preparing a set of quizzes for student use. A more detailed work-flow is giving in figure 11.