Analysing Learner Behaviour through Logfiles for Quality Assurance and Student Evaluation

Ralf Hauber

Telecooperation Group at the Department of Computer Science, University of Linz, Austria

Abstract

Learning environments provide extensive logging capabilities that document learner activities. Careful interpretation of these logs can be used for quality assurance and student evaluation. We present a logfile processor that reads Toolbook logfiles and collects data for determining "hot spots" (i.e. particularly good or bad spots) in a course or in learner behaviour. Hot spots show up as extreme values during the analysis; they are candidates to be looked at by instructional design experts. The logfile processor's methods for data collection can be customised using an event mechanism. Its output — either flat ASCII or structured XML — is further analysed: We present exemplary diagrams based on logfiles from a sample course.

Key words: Learner behaviour, data collection, data analysis, data visualisation, quality assurance, student evaluation.

1 Introduction

Consider a general e-Learning situation: There is a learner, and there is a course. The learner works through the course. Afterwards, *if* she has learned, *something* in this situation has *good quality:* the learner, the course, or both.

Quality assurance

Quality assurance systematically *monitors* and *evaluates* various aspects of the learning process to ensure that *standards of quality* are being met. There are various catalogues of criteria for making more concrete what "standards of quality" could mean (for instance with respect to instructional design, user interface, or reuse of implementation), and most of those criteria need a human expert for evaluation: For a course, experts in instructional design answer the questions "Is the course good (or bad)?" and "*Why* is the course good (or bad)?". The former question is important since usually much money was spent to have the course developed, and the latter to produce more good and less bad courses in the future.

Using automatically logged data to cut down the experts' search space

"Hot spots" are particularly good or bad spots that can be found by analysing learner behaviour. Indications for hot spots are, for example, a chapter with many jumps to the glossary, or a chapter that is visited only for a few seconds, or a question that is never answered correctly. We strive to cut down the instructional design experts' search space for hot spots in courses and in learner behaviours by analysing logfiles. After the analysis, the experts get a list of hot spots that are suggested for further examination. One good thing about this approach is that the logged data is available for free — the logging feature just has to be turned on in the learning environment.

Processing and analysing logfiles

We present a logfile processor for Toolbook [1] logfiles that collects data for further analysis. These data reflect learner behaviour and are used to draw conclusions regarding the quality of the course and the learning process. The logfile processor provides an event mechanism to easily add further data collection methods. Currently we collect (1) total learning time; (2) time spent per chapter; (3) time spent per question; (4) time spent within a chapter before jumping to another; (5) sequence of visited chapters; (6) number of jumps to a chapter; (7) how many and which answers were correctly, partially, falsely, or not answered. Note that the Toolbook logfile is not quite sufficient for most of these analyses since it is important to know the "units of interest" for which a particular analysis is carried out. Currently the unit of interest is either a "page" or a "chapter" (see Section 2.3).

Structure of the paper

Section 2 describes the logfile and how it is processed by a parser that fires events. Section 3 uses the parser's events to implement several data collection methods that provide data for further analysis. Section 4 assesses the approach taken, reports on the status of implementation, and contains remarks and related work. Section 5 summarises.

2 Logfile processing

This section describes the structure of Toolbook logfiles and the design of the logfile parser. The logfiles contain entries for navigation, user input, and grading of user input.

2.1 Structure of logfiles

A Toolbook logfile is a text file that consists of three sections.

i. The *header* contains general information about the learning session (learner's name, filename, time, and date).
ii. The *list of actions* documents the learner's activities (traversal to another page, and user input; for some reason the total score of a quiz is also included in this section).
iii. The *report of answers* grades and reports on the answers that were given in a quiz.

Figure 1 shows a portion of a logfile. The *header* is separated from the *list of actions* by several dashes. An *action* always starts with a timestamp, and by convention (see Section 2.3) the chapter number precedes the page title. In Figure 1 only some log entries for chapters 1 and 7 are shown, and the chapter number is emphasised using a bold type style. The last section, the *report of answers*, starts after the line "`SESSION END...`". The logfile is slightly edited for readability, omissions are marked by "`[ . . . ]`".

```
11:47:48 : Francis          2000 03   20
First Aid  C:\Courses\Analyzer Test\firstaid.tbk
------     ------     ------
11:47:49   Page Acknowledgement
11:48:14   Page TitlePage
11:48:19   Page Contents
11:48:25   Page 1 Introduction
11:48:28   Page 1 1/6 CivicDuty
[...]
12:03:02   Page 7 Quiz
12:03:03   Page 7 1/13 Administering
12:03:06   7 Administering R/F              "yes"
12:03:09   Page 7 2/13 Helper
12:03:21   7 Helper MC                      "analysing"
12:03:22   7 Helper MC                      "planning"
12:03:23   7 Helper MC                      "acting"
[...]
12:04:17   Page 7 5/13 DangerZone
12:04:29   7 DangerZone FITB                "<100> "100""
12:04:30   Page 7 6/13 Helmet
12:04:38   7 Helmet DAO                     ""
[...]
12:10:12   7 HeartMassage FITB1             "<2> "2""
[...]
12:10:19   Page 7 Score
12:10:25   Q=Score Quiz : E=ShowScore! : S=12.32,16
12:10:34   Page End
SESSION END      SUMMARY    00    22    59
Question            Score  Max Score Locked  Tries used Max  [...]
7 Assistance R/F      1     1         FALSE   1          0    [...]
7 Helper MC           1     1         FALSE   1          0    [...]
7 RescueTeam MC       1     1         FALSE   1          0    [...]
[...]
```

Figure 1: A portion of a logfile (slightly edited).

2.2 Parsing a logfile

The parser reads the logfile line by line. Each line is processed, and a corresponding event is fired.{1} Events are sent to listeners that are registered with the parser.{2} Following the structure of a logfile (three sections: header, list of actions, report of answers), there are three kinds of listeners. Each listener may receive a specific event. Figure 2 lists the corresponding sections, listeners and events.

section	listener	event (top-level)
header	`SessionInfoListener`	`SessionInfoEvent`
list of actions	`ActivityListener`	`ActivityEvent`
report of answers	`ScoreListener`	`ScoreEvent`

Figure 2: Sections, their listeners, and the listeners' top-level events.

The events from Figure 2 are specialised (refined); therefore the heading of the last column says "top-level". In the analysis (Section 3), the specialised events are used to pick out exactly the information needed.{3} Figure 3 lists the specialisation of the events.

event	specialisation	meaning
`SessionInfoEvent`	`SessionDataEvent`	General information about the learning session.
	`TimeSummaryEvent`	Total time of the learning session.
`ActivityEvent`	`TraversalEvent`	The learner navigated to a different page.
	`InteractionEvent`	The learner made some input (in a quiz).
	`ScoreSummaryEvent`	The total result of a quiz (score vs. maximum score).
`ScoreEvent`	(none)	Detailed report on the learner's input.

Figure 3: Specialisation of events.

2.3 Grouping pages

Usually an analysis (Section 3) is done for certain *groups* of pages, not for a single page. The most intuitive grouping probably is "by chapter": Each chapter in a course defines a set, which contains all pages that belong to that chapter. More advanced grouping criteria might involve categorization such as "all pages that give general surveys" or "all pages that contain a simulation".

We group pages using a naming convention for the (Toolbook-internal) title of a page: The title always

starts with the chapter number. This was already shown in the sample logfile in Figure 1. As long as there are only very few grouping criteria involved, this works fine.

3 Logfile analysis

Using the events provided by the logfile parser (Section 2), we implemented eight classes for basic analysis. A *basic analysis* considers only a *single* logfile. But, since the goal is to give hints to human experts who should be relieved from examining every single learning process, a single logfile is of little help.{4} Hence the results from *several* logfiles are combined and further analysed using methods from descriptive statistics. We use the term *combined analysis* because it combines the results from several basic analyses. Note that in this paper *basic analysis* is essentially *data collection* from logfiles.

3.1 Basic analysis

Basic analysis uses the parser's events to collect data from the logfiles. Typically in each analysis only one kind of event is of interest. As an example, Figure 4 shows the code for computing the *sequence of chapters* a learner has visited. Since the needed traversal information is part of the *list of actions*, an `ActivityListener` is implemented. The `ActivityEvents` have to be handled in the method `processActivityEvent`; inside this handler, only `TraversalEvents` are of interest. For storing the sequence of chapters there is a class attribute `chapterSequence`. On every change of chapter the handler adds the new chapter to the sequence.

```
// code excerpt from class ChapterSequence (essential method)

LinkedList chapters = new LinkedList();    // chapter sequence

public void processActivityEvent( ActivityEvent e) {
  if( e instanceof TraversalEvent) {
    TraversalEvent te = (TraversalEvent)e; // typecast the event
    String chapter = te.getChapterId();    // unique chapter id

    if( chapters.isEmpty())          // is it the first entry?
      chapters.addLast( chapter);    //   yes: add the chapter
    else // no: not first entry (chapters are in the sequence)
      if( ! chapter.equals( chapters.getLast())) // changed?
        chapters.addLast( chapter); //   yes: add new chapter
  }
}
```

Figure 4: Computing the sequence of visited chapters.

Passing data from basic analysis to combined analysis

The goal of *basic analysis* is to provide data for *combined analysis*. Therefore there must be a defined data format for passing on the results. Originally the results of the basic analysis were stored in a flat ASCII file, which then was imported into Microsoft Excel. Later on, the ability to write XML was added. This was done for two reasons: (1) It is easier to examine the hierarchically structured XML file, and (2) it is easier to switch the tool for combined analysis. Figure 5 and Figure 6 show the signatures of the corresponding methods `toString` and `generateXML`, and some sample output.

```
// ASCII output is simply generated by the toString method
public String toString();

// sample ASCII output
chapter    seconds
1          162
2          158
3          58
4          152
[...]
```

Figure 5: ASCII output of the basic analysis.

```
// XML output is written to a Writer (from package java.io)
public void generateXml( Writer out);

// sample visualisation of xml output (e.g. in internet explorer)
- <analysis_result>
  - <time_per_chapter>
      <!-- enties with chapter and seconds -->
    - <entry>
      <chapter>1</chapter>
      <seconds>162</seconds>
      </entry>
    + <entry>
      [...]
    </time_per_chapter>
- <chapter_sequence>
    <chapter>Acknowledgement</chapter>
    [...]
  </chapter_sequence>
  [...]
```

Figure 6: XML output of the basic analysis.

Implemented classes for basic analysis

In the remaining section some basic analyses are presented.{5} Their implementation is a little more complex than the example in Figure 4, but not very different. The structure of the following presentation of basic analysis classes is: «Description of the data to be gathered. (`listener class name: implemented interface name`) `Sample output`».

- Total learning time. (`TimeSummary: SessionInfoListener`)
  `<reported_time_summary>00:22:59</reported_time_summary>`
- Time spent per chapter. (`TimePerChapter: ActivityListener`)
  `<entry><chapter>1</chapter><seconds>162</seconds></entry>`
- Time spent per question. (`TimePerQuestion: ActivityListener`)
  `<entry><question>resuscitation</question><time>9</time></entry>`
- Time spent within a chapter before jumping to another. (`TimeBeforeLeave: ActivityListener`)
  `<entry><chapter>4</chapter><seconds>248</seconds></entry>`
- Sequence of visited chapters. (`ChapterSequence: ActivityListener`)
  `<chapter>Acknowledgement</chapter>`
  `<chapter>Introduction</chapter>`
- Number of jumps to a chapter. (`JumpsToChapter: ActivityListener`)
  `<entry><chapter>2</chapter><jumps>5</jumps></entry>`
- Summary of the score. (`ScoreSummary: ActivityListener, ScoreListener`)

```
<reported_score_summary>
  <points>12.32</points>
  <maximal_points>16</maximal_points>
</reported_score_summary>
<computed_score_summary>
  <questions>16</questions>
  <correct_answers>10</correct_answers>
  <partially_correct_answers>4</partially_correct_answers>
  <false_answers>1</false_answers>
  <not_answered>1</not_answered>
</computed_score_summary>
```

- Detailed report on the score. (`CorrectAndFalseAnswers: ScoreListener`)

```
<correct>
  <score_entry>
    <chapter>7</chapter><name>breathing</name><score>1</score>
  </score_entry>
</correct>
<partially_correct>
  <score_entry>
    <chapter>7</chapter><name>heart</name><score>0.67</score>
  </score_entry>
</partially_correct>
<false>
  <score_entry>
    <chapter>7</chapter><name>pulse</name><score>0</score>
  </score_entry>
</false>
<not_answered>
</not_answered>
```

3.2 Combined analysis

In this section some candidates for combined analysis are presented. They all were applied to data from a sample course that was worked through by 16 learners. (The results are not being interpreted in this paper. For a comment on the general interpretation problem see Section 4.2.1.)

3.2.1 Total learning time

The total learning time is a coarse measure of the learner's effort. Descriptive statistics provides a condensed description of the data: minimum, maximum, median, standard deviation, quartiles. Figure 7 shows a bar chart of the total learning time for the 16 learners.

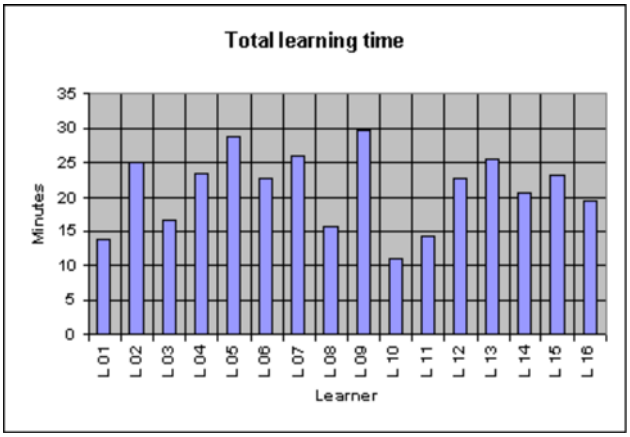

Figure 7: Total learning time for 16 learners.

We also correlated the total learning time and subjective learning time. (The learners had to fill in a questionnaire immediately after the course. One of the questions contained a subjective assessment of the learning time.) Another analysis correlated the total learning time and the results of the quiz.{6}

3.2.2 Time spent per chapter

The average time a learner spent per chapter is computed. In Figure 8, for each chapter the mean, median and a standard deviation interval (= mean ± std. dev.) is shown.{7} The lines are smoothed for better perception (though the axes have a discrete scale). The chapters "acknowledgement", "contents", "title", and "help" took almost no time; chapter 7 was the quiz.

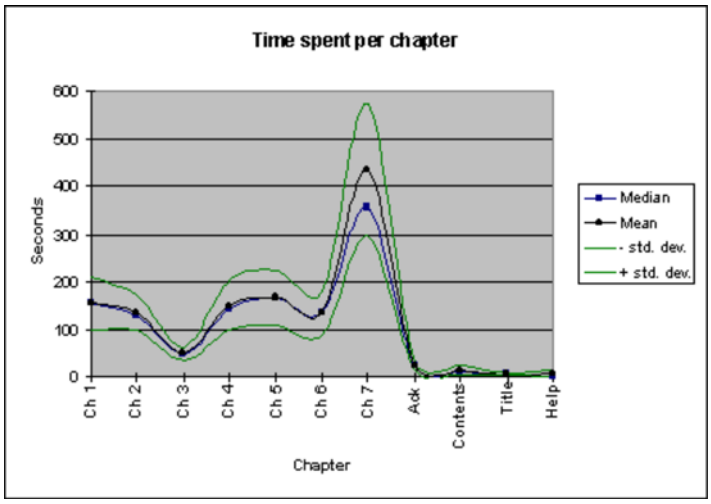

Figure 8: Time spent per chapter.

3.2.3 Sequence of chapters

The sequence of chapters shows how a learner navigated through the course. In Figure 9 the course was navigated rather sequentially. Only in the first eight minutes three jumps to "contents" and one to "help" occurred. (For a comment on a different style of navigation see Section 4.3.)

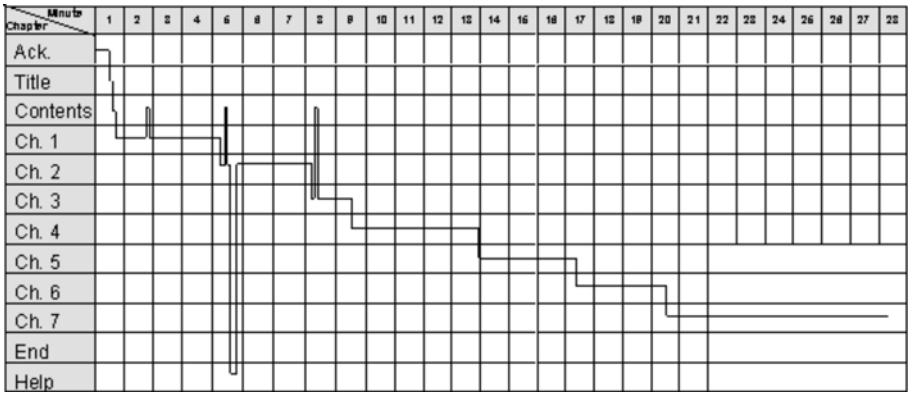

Figure 9: Sequence of chapters. Translated from [2].

3.2.4 Jumps to a chapter

The jumps to a chapter can be read off the sequence of chapters Figure 9). They indicate how frequently a chapter was visited. Since a learner might stay inside a chapter for a longer time, the time spent per chapter (Section 3.2.2) should be examined in parallel.

The diagram is not shown since it looks similar to Figure 7 with "chapters" on the x-axis and "number of jumps" on the y-axis.

3.2.5 Quiz-related analysis

a)

The "total answering time for the quiz" can be singled out and displayed like the total learning time in Figure 7.

b)

The "time for answering questions" shows the average answering time for each question. The resulting diagram is similar to Figure 8 with "questions" on the x-axis.

c)

The "total score" is similar to Figure 7 with the score (absolute or percentage) on the y-axis.

d)

The "correctness statistics" reports on the number of totally correct, partially correct, falsely answered, and unanswered questions. Figure 10 is a diagram showing the portion of those four "correctness classes" per question (alternatively it could be done per learner). To compare correctness classes *among* questions, Figure 11 is better suited (it should be sorted by the values of the correctness class in question).{8}

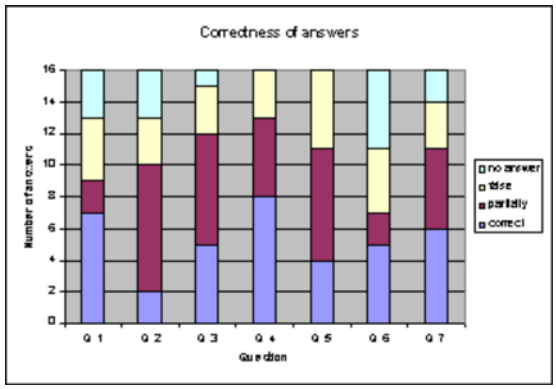

Figure 10: Cumulated correctness of answers (per question).

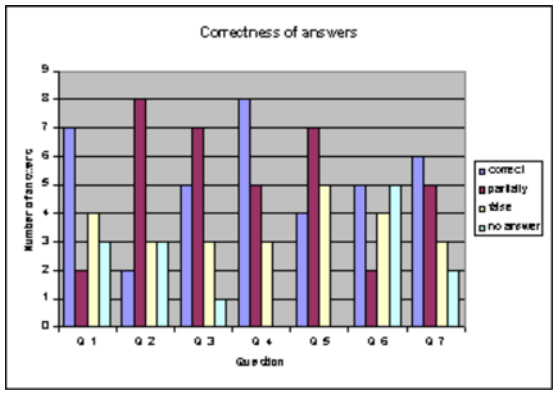

Figure 11: Correctness of answers (per question).

4 Assessing the approach, status of implementation, and related work

4.1 Implementation and validation

The Java implementation of the logfile processor was pretty straight forward. The only awkward thing was that the structure of the logfile was not precisely described in the documentation. For parsing, Java's `StringTokenizer` class was sufficient. As anticipated the naming convention for grouping pages (Section 2.3) proved limiting in practical use.

To get a feeling for the logged data, 16 students worked through a small example course. The produced logfiles were used to test the logfile processor (parsing, event mechanism, output). The logfile processor's output was used to examine combined analysis (Section 3.2). The newer XML output scheme (Section 3.1) was only used for browsing the output. For most analyses we used Microsoft Excel, which imports plain ASCII.

4.2 Notes on the analysis

In this section some comments on the analysis (Section 3) are made.

Added value of the combined analysis

Basic analysis is trivial in the sense that it is an application of standard techniques. It is the *combined analysis* that adds real value for quality assurance and student evaluation. There are two levels of added value:

1. *Modest level:* The extreme values found during analysis indicate hot spots that are to be further analysed by instructional design experts. (So far we operated at this is level.)

2. *Advanced level:* Beyond just tracing extreme values, the learner model and the presented content are included in the analysis. (This kind of data is modelled in advanced learning systems anyway. Extensive field evaluation is necessary to assess whether and how results from combined analysis let instructional design experts deduce parameters describing navigation and orientation within a course, learning style, speed of learning, or success of learning.)

Graph-based analysis

One kind of analysis not shown is graph-based analysis. This is mentioned because it is well suited to visualise results from combined analysis. Colour, size, shape, and neighbourhood of graph items can be used for a concise presentation of results.

4.2.1 Interpretation of analysis results

Any interpretation of combined analysis beyond "modest level" has to be done with great care. For example the seemingly simple comparison of total learning time (Section 3.2) cannot be interpreted without further knowledge about the whole learning situation. A short learning time may mean that the foreknowledge was excellent, or that the learner gave up early, or that the learner believed she already knew the content, etc. Further if the foreknowledge was excellent, is it because of a change in the syllabus, or because the learner repeats the course?

Sometimes it might suffice to take a combined look at the total learning time and the final score. But the solution to this diversity of interpretations is a general one, and probably too obvious to the reader to be explicitly mentioned: Be clear about the learning situation,{9} and about the kind of statements that should be made based on the analysis.{10}

Two applications of interpreted logged data are (1) Schaper [4], who uses logfiles among other evaluation techniques to validate his co-operative learning environment, and (2) computer-managed instruction (CMI), where data gets collected, interpreted and used to improve the quality of computer based learning (CBT). An AICC document on CMI describes the data flow from the CBT system to the CMI system: "The CMI system reads the CBT-to-CMI file, updates applicable student data, and determines the next student assignment or routing activity." [5]

4.3 Navigational strategies

The *sequence of chapters* (Section 3.2.3) can be used to determine the "degree of exploratory behaviour" by using a distance measure (distance from an "intended navigation"). [4] Our example course was designed rather sequentially, so – not surprisingly – a rather sequential navigation pattern was observed. But even here one learner used a totally different pattern: She started with the quiz, and for each question that she could not answer she jumped to the table of contents, from there to the chapter where she expected the answer, and then back to the question in the quiz.

A detailed sequence analysis may also indicate missing links in the course, and show from which page (or chapter) the help is used frequently — probably a page that should be improved.

For a different use of a sequence of pages (i.e. a *trail*) see [6]: Trails provide a mechanism that allows learners to ask questions such as "where do other learners go from here", "what else should I read?" or "how did we come to that conclusion?". The fact that trails are built with information about the users' browsing paths and activities makes them well suited for collaborative applications where users with similar interests are to be matched.

Describing structure and navigation in hypermedia is a goal of the GUTS project on typing graph-based structures. This project provides the greater context of this paper. Some of its structure-related issues regarding the authoring of courses are discussed in [7].

4.4 Future scenario

The introduction mentioned the idea of using solely *automated* data collection. The logfile is just one such data source. Other parameters may be considered:{11} pulse rate, blood pressure, absence from the course (motion tracking or video input), eye tracking, stress in muscles, neural activities, etc.

This may sound a little far fetched, but as long as computer based learning is not thoroughly understood and as long as learners do not learn effectively, every attempt that might improve learning should be pursued. To make lifelong learning a lifelong pleasant experience.

5 Summary

We presented a processor for collecting data from Toolbook logfiles. The logfiles are generated automatically, and the sole overhead is to define the "units of interest" for the analysis (we used the chapter structure). Eight data collection methods were shown. New data collection methods can easily be added using an event mechanism.

The output of the logfile processor (flat ASCII or structured XML) is further analysed to determine hot spots in a course or in learner behaviour. Those hot spots are candidates to be looked at by instructional design experts. A couple of *combined analysis* approaches were shown and applied to data from a sample course.

References

[1] Toolbook Web page. http://www.click2learn.com/

[2] Trunk, Petra: *Automatische Qualitätsbewertung für e-Learning (Automated quality assurance for e-learning)*. Master thesis, Polytechnic University of Upper Austria, Department of Media-Technology and -Design, Hagenberg, Austria, 2000. In German.

[3] Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[4] Schaper, Joachim: *Lebenszyklusunterstützung in kooperativen Lehrsystemen (Supporting lifecycle in co-operative teaching systems)*. PhD thesis, Department of Computer Science, University of Karlsruhe, Germany, 1995. In German.

[5] AICC: Document AGR006 on *Computer Managed Instruction (CMI)*, Version 2.0, 1998. http://www.aicc.org/docs/AGRs/agr006v2doc.zip

[6]
Reich, Siegfried; Carr, Leslie; De Roure David; Hall, Wendy: *Where have you been from here? Trails in hypertext systems.* ACM Computing Surveys 31, 4, Dec. 1999.

[7]
Hauber, Ralf: *Improving Authoring by Enforcing Reuse of Structure.* International Conference on Information and Communication Technologies for Education (ED-ICT), Vienna, Austria, 2000. Accepted for publication.

Author

Ralf Hauber, Dipl.-Ing.
Telecooperation Group
Department of Computer Science, University of Linz
Altenbergerst. 69, A-4040 Linz, Austria
ralf.hauber@jk.uni-linz.ac.at

Acknowledgements

The author thanks Dipl.-Ing. (FH) Petra Trunk, who developed the example course and conducted and analysed the survey while working on her master thesis at the Department of Media-Technology and -Design of the Polytechnic University of Upper Austria, Hagenberg, Austria.

Footnotes:

{1}
Apart from a minor exception there is exactly one event per line.

{2}
This is known as the Observer/Observable design pattern. [3]

{3}
There was some debate whether we should choose more listeners and not specialise events, or less listeners and more specialised events — or maybe abandon the listeners at all and use a Builder design pattern [3]. Implementing the analysis classes, we felt that the current choice is a fair trade off.

{4}
A single logfile might be of interest if it strongly deviates from the average logfile. And, of course, when evaluating a specific student.

{5}
In addition there are three “debug” listeners (one for each listener interface shown in Figure 2 that just write a protocol containing the events received.

{6}
This can also be done on a per-chapter basis.

{7}
This can also be done on a per-learner basis.

{8}
The average score per question would be a useful additional information in Figure 10 and Figure 11.

{9}
Different things hold for primary school, high school, university, continuing education, etc.

{10}
Examples: Is it to pay a grant, or to rework the syllabus, or to evaluate the course?

{11}
Those need specialised hardware and are more visible (if not hindering) to the learner.