ELSEVIER

Information and Software Technology 39 (1997) 541–549

INFORMATION AND SOFTWARE TECHNOLOGY

Code-based analysis of the development effort of a large scale courseware project

I.M. Marshall[a,*], S. Price[b], P.I. Dugard[c], P. Hobbs[b], W.B. Samson[c]

[a]*Department of Information Technology, University of Lincolnshire and Humberside, Cottingham Road, Kingston upon Hull HU6 7RT, UK*
[b]*Centre for Computing in the Social Sciences, University of Bristol, Bristol, UK*
[c]*School of Informatics, University of Abertay, Dundee, UK*

Received 11 March 1996; accepted 27 February 1997

Abstract

This paper investigates proposed code-based measures for multimedia courseware as part of an on-going program of research into effort estimation models. The proposed code-based size measures are analysed using source code and project data from WinEcon courseware written in the ToolBook authoring language. Statistical analysis of a priori, ToolBook specific and portable subsets of 47 code measurements are presented along with proposed predictive models of courseware development effort based on a priori, best subset and portable subset modelling assumptions. © 1997 Elsevier Science B.V.

Keywords: Measures; Multimedia; Effort estimation

1. Introduction

Multimedia courseware production is a growth industry in the current climate of educational expansion combined with increasing constraints on all public spending. However, there is plenty of evidence that the huge investment being made in many countries on courseware production frequently results in products which are over budget, are not delivered on time or, alternatively, have reduced media content and learner interactivity to deliver the product on time and within the planned budget [1–4]. It is clear that the research effort previously made in software engineering now needs to be undertaken in courseware engineering to improve the productivity of the development process and quality of the final product or to estimate development effort and time [5]. This paper presents an analysis of the relationship between code-based measures and development effort associated with a large scale courseware project.

1.1. The size problem

Marshall et al. [4] have exposed some new problems related to courseware development not previously encountered in software engineering. One such problem is measuring the size of the product [6,7]. In software engineering, source lines of code [8], function points [9], object counts [10] or object point measures [11] have been used as size measures. In multimedia courseware the use of existing software size measures may be inadequate because the development of program code to integrate media elements can represent a small proportion of the total development effort [12]. The development and integration of video, animation and graphical elements of courseware can swamp the coding effort [13]. From research carried out in software engineering, there is also some doubt as to the suitability of traditional measures such as source lines of code or function points for the 4GL or object-oriented languages normally used in courseware development [14,15]. The most commonly used size measure in the multimedia courseware development industry is 'learner-time', which represents the time a 'target learner' would spend working through the material [4]. This is of limited usefulness because of the vagueness implied in the term 'target learner'. Which one? The average? The slowest? In any case, there is no evidence that any target learners are ever timed so this measurement appears to be the learner time the client requested at specification or estimated at delivery [6,7]. While there is widespread agreement that learner time is not ideal [16], it does have the advantage of being widely

* Corresponding author. Tel.: +44 1482 440550; fax: +44 1482 442393; e-mail: i.marshall@staff.humber.ac.uk

used by commercial developers and is easily understood by clients.

1.2. Alternatives to learner time

A number of alternative measures to learner time have been suggested as courseware size measures such as screens, performance objectives, media objects, interactions, lessons, code-based measures and conventional teaching time compression [7,17,18]. A more detailed analysis of the advantages and disadvantages of each of these potential design-based measures are beyond the remit of this paper and will be presented in a later publication [17]. However, a brief discussion of the one of the proposed measures is presented here to indicate the types of problems encountered.

A size measure closely related to 'learner time' is sometimes used: the conventional teaching time that the courseware will replace. One problem with this is that the kind of teaching is not usually specified: it could be lectures, seminars, tutorials or workshops or some combination of these. Experience with courseware suggests that many students cover the ground faster using courseware than with conventional teaching. Claims of a 'compression' to as little as 16% conventional teaching time have been made [19], but a compression to 70% conventional teaching time is considered normal by authors who have conducted meta-analysis of existing research [17,20]. Even if conventional teaching time could be worked up into an adequate size measure now, its life would be limited if we see the expected large expansion of courses delivered by means of courseware. Many will not be replacing existing conventional courses and developers and educators will soon lose touch with conventional equivalents.

1.3. Size is just the beginning

The problem of finding a good size measure has a knock-on effect on the problem of estimating effort and development time [5]. Measures or estimates of size and/or complexity are fundamental to attempts to estimate development effort of software products. While size measures have at least been considered for courseware development, measures of complexity, maintainability, usability and other aspects of product quality have scarcely been discussed [6]. In addition to the largely unconsidered area of measuring the technical quality of the product, there is the non-trivial problem of measuring its educational effectiveness [17].

All this indicates that there is considerable work to be done in identifying courseware measures which may help to quantify some of the features of interest. Considering the importance of the multimedia courseware development industry, it is perhaps disappointing that only limited research has been carried out to measure its products and processes. However, it is only through developing a better understanding of the courseware measures that more accurate predictions of effort and development time can be made. This study is a contribution to this work in which WinEcon, a large scale courseware development project, is used to investigate the relationship between learner time, potential code-based measures and effort estimation models.

1.4. Overview of WinEcon

Briand et al. [21] suggested that in order to "...validate assumptions experimentally, one can adopt two main strategies (1) small-scale controlled experiments (2) real-scale industrial case studies." In this investigation the opportunity to use the second alternative was made available through the involvement of the WinEcon project management team. The WinEcon courseware project was funded under the United Kingdom Higher Education Funding Council's (HEFC) Teaching and Learning Technologies Programme (TLTP) [22,23]. Eight Universities collaborated and the project was co-ordinated by the Centre for Computing in Economics at Bristol. The aim was to produce Microsoft Windows™ based materials to replace tutorials and augment lectures for students on 'Introductory Economics' courses [24]. The resulting 163 346 lines of source code provided 25 chapters organized as a total of 693 pages. The intention was that each page would provide up to 20 min of learner time and that each chapter would typically provide 3 to 5 h of student activity. WinEcon makes extensive use of text, graphics, simulation and student interaction to deliver the material. To meet the requirements of individual lecturers, WinEcon is also designed to be easily customised by means of the WinEcon Lecturer program. In the version used as the basis of this analysis there are no audio or video elements although these may be introduced in future releases of the courseware. A fully integrated student record keeping system has recently been completed.

The WinEcon project core development team had an experienced software development manager, and was responsible for project co-ordination, overall design and the provision of development tools. The development of individual chapters was the responsibility of authors and developer teams. Authors were experienced undergraduate economics educators who had varying levels of experience of courseware production. Each of the developer teams contained a mixture of experienced and inexperienced developers who had not previously worked together. The newly formed consortium had no tools, templates or house styles before the start of the project. These were developed as the project progressed and are now available as a resource for future developments.

WinEcon courseware was developed in Asymetrix ToolBook™ over a period of 3 years between 1992 and 1995. ToolBook is an authoring language which uses a book and object metaphor as the basis for coding [25,26]. Pages form the basis of the high level structure while objects, including pages, can have OpenScript® programs attached. ToolBook claims to be object-oriented in nature, but this appears to be limited to the ability to handle events and to attach programs to pages and other objects. The finished courseware is delivered using the ToolBook Runtime which runs the software but prevents editing of pages or OpenScript programs. During the project ToolBook was upgraded from version 1.5 to version 3. This required considerable reworking of the original code to take advantage of built-in features which had previously required specially developed functions.

The WinEcon software is provided at nominal cost to Higher Education Funding Council supported institutions in the United Kingdom, where it will be used by a target population of approximately 12 000 students per academic year. In addition, the material will be sold to commercial organisations in the UK and to both commercial and higher education establishments overseas.

2. Data collection

Measures were collected from the code using a Toolbook-based automated tool, and work records were used to quantify the effort expended [12]. Data collection began when the WinEcon project team had completed the first public release of the courseware and had just started to work on the second. While being aware of the potential dangers of being accused of a 'shotgun correlation approach' [27] the automated tool collected measures from the source code specifically for effort prediction while some others are to be used as part of the ongoing project. In total 52 measures, all counts of code-features, were produced by the automated tool. All the measures were grouped under the following three main headings:

- WinEcon specific measurements;
- ToolBook specific measurements;
- Portable measurements.

The first group of five WinEcon specific measurements are unique to this development. They represent coding constructs developed as part of the core development team's overall design philosophy. As such, they are unlikely to be useful outside the consortium and they are not described here. The remaining 47 measures are presented here for completeness before those collected for effort estimation purposes are discussed along with modelling assumptions.

2.1. Toolbook specific measurements

The second group of ten ToolBook specific measurements describe OpenScript language or page metaphor specific features. They are described in ToolBook terms but they could be extended to other authoring systems or languages with suitable adaptation. Table 1 describes the ToolBook measures.

Table 1
Description of the ten ToolBook specific measurements

Measurement	Description
User-properties	User defined variables or data items attached to the page.
Captioned-buttons	Buttons of any type with a textual caption.
Groups	A collection of other objects grouped into a single entity.
Level-one-objects	Objects excluding objects nested inside groups.
Scripted-objects	Objects with OpenScript code attached.
Propertied-objects	Objects with user properties such as variables or data items attached.
Notify-objects	Objects having a 'demon'. A 'demon' is a vectored or hooked message handler.
To-sets	Assignment function definitions.
Notify-handlers	'Demon' vectored or hooked message handling definitions.
Pagescript-raw-lines	Raw lines including blank lines and comment lines in the script of the page object itself.

2.2. Portable measurements

The final group of 37 portable measurements describe features which are not ToolBook specific. They are described in OpenScript terms which are similar to other authoring systems or programming languages. Table 2 describes the portable measures.

Project records and questionnaires were used to collect data about development effort, development time, team size and other information [6] for each of the chapters produced by each team. Table 3 lists the teams size at the start of the development process, development effort, full time equivalent team size for each of the WinEcon chapters. The WinEcon chapter numbering system is used in Table 3. Chapter 9 is missing from the list because it was originally planned but eventually amalgamated into other chapters during implementation.

3. Analysis of data

Predicting effort, and hence cost, is one of the major concerns of most measures programs. However, so far attempts to predict effort for courseware projects have not been very successful [28], so anything that can be learned about prediction from this project will be very valuable. There are 47 measures and if we just look for those whose correlations with effort are significant at the 5% level we might expect to find 2 or 3 by chance even if none of the measures would have significant correlations with effort in the wider population of all courseware developments. In fact there are 15 correlations significant at the 5% level, which gives some grounds for supposing that potentially useful relationships are there to be investigated. At this early stage in our understanding of the courseware development process "We need a large body of careful measurements for the software community to determine what

Table 2
Description of the 37 portable measurements

Measurement	Description
Pages	Pages associated with each module.
Objects	Objects on a page such as button, field, line group, page or book. This includes objects nested inside groups.
Graphics	Bitmap graphic objects excluding ToolBook lines or ellipses, rectangles etc.
Fields	Static and editable text fields. Sometimes these will not have text in them because they are used to produce an attractive insert or raised bevel effects on the screen.
Static-fields	Non-editable text fields. These may be made editable at run-time by OpenScript coding.
Edit-fields	Editable text fields. These may be made static at run-time by OpenScript coding.
Listbox-fields	Selection lists.
Scrolling-fields	Fields with a scroll bar.
Words	Words in all the text of all fields of any field type. This includes all text of WinEcon text card fields even though the text displays only one line at a time.
Characters	Characters in all text of all fields of any field type. This includes all text of WinEcon text card fields even though the text displays only one line at a time.
Buttons	Buttons of any type. WinEcon often uses disabled, borderless buttons as a place holder for bitmap graphics.
Push-buttons	Push buttons.
Radio-buttons	Radio buttons. These are used for exclusive selections.
Check-buttons	Check buttons. These are used for non-exclusive selections.
Graphic-buttons	Buttons of any type which have a bitmap graphic.
Combo-boxes	Drop down editable or static selection lists.
Name-objects	Objects having a mnemonic name. This is usually to allow referencing to it from OpenScript code.
Raw-lines	Lines in all scripts of all objects including blank and comment lines.
Blank-lines	Blank lines (white space lines) in script of all objects.
Comment-lines	Comment lines but excluding trailing comments at the end of lines.
Code-lines	Lines of actual code excluding white space lines in scripts of all objects. Since statements may be split over multiple lines this is not a count of OpenScript statements.
ifs	'if' statements in scripts.
elses	'else' statements in scripts.
conditions	'condition' statement in scripts. This is similar to the 'switch' in C.
whens	'when' statements in scripts. This is similar to the 'case' statement in C.
steps	'step' statement in scripts. This is similar to the 'for' in C.
whiles	'while' statements in the script.
dos	'do' statements in the scripts.
sends	'send' statements in scripts. This is similar to procedures in C.
to-gets	Function definitions.
to-handles	Handler definitions message.
forwards	Message forward statements.
breaks	'break' statements which are used to break out of loops, conditions/whens and exit handlers by quitting.
continues	'continue' statements.
systems-variables	System or global variables.
Local-variables	Local variables.
Trailing-comments	Lines of code with trailing comments.

relationships actually exist. This information is not available because it is not collected, or if it is collected it is proprietary'' [27]. Analysis involved attempting to produce equations for predicting effort using multiple regression of effort on subsets of the available measures. Where potentially useful regressions were identified, graphical plots were used to check that the assumptions of the regression model were not violated.

The subsets were defined in three ways: by a priori considerations, attempting to find the best predictors, and the best portable predictors. The relationships among the size measures were also explored. In the following analysis correlations among group of variables and correlations between effort and some of the variables are considered. Correlations is being used as a means of selecting potentially useful subsets of variables by drawing attention to those that exceed the critical values for significance at the 5% level. In the conventional sense testing the hypothesis that the correlations are zero was not the prime interest. As a vehicle for identifying potential useful subsets for further analysis the choice of the 5% level of significance was found to provide subsets of manageable size without excluding too many potentially useful variables.

3.1. Predicting effort using a priori considerations

Research from software engineering measures indicated that the size, complexity and quality of the product influenced the effort consumed in its production [10]. Four of the measures relate to size; *raw-lines* of code, number of *objects*, number of *named-objects*, and number of *pages*. Table 4 gives the correlations among effort and the size

Table 3
Summary of development data collected from WinEcon project teams

WinEcon chapter number	Development team number	Development effort (developer h)	Full time equivalent team size
1	4	350	1.25
2	4	672	1.50
3	4	700	1.25
4	5	3360	1.50
5	1	1225	1.25
6	1	1225	1.25
7	3	2100	1.25
8	5	1890	1.50
10	7	1050	1.25
11	8	1680	1.50
12	2	1470	1.50
13	2	1470	1.50
14	2	1470	1.50
15	7	1400	1.25
16	8	1260	1.50
17	7	1400	1.25
18	8	2184	1.30
19	3	2100	1.25
20	1	1225	1.50
21	8	420	1.50
22	5	3150	1.50
23	1	700	1.25
24	3	1050	1.25
25	6	1050	1.25
26	6	1050	1.25

measures. All are significant at the 5% level. The number of *pages* could be used to provide an estimate of learner time, since it was the developers' intention that each page would occupy the students for about 20 min.

There are four traditional measures related to complexity [29]; *ifs, elses, conditions,* and *whens*; other measures related to complexity which may be more appropriate in this context are number of *propertied-objects*, *WinEcon-graphs* and *steps*. Table 5 gives the correlations among effort and the complexity measures, with those which are significant at the 5% level highlighted in bold. The variations in quality were thought to be rather small because of the way the project was managed, and we did not attempt to quantify them.

A stepwise regression on these 11 measures suggests that only *pages* and *elses* make a significant contribution to prediction, and the regression on these two accounts for only 55% of the variance of effort. Though the regression is highly significant it is unlikely to give useful predictions. The difference between statistical and practical significance is not always appreciated. A regression is statistically significant if the correlation between observed and predicted values of the dependent variable is significant. For 25 observations a correlation of less than 0.4 is statistically significant but it would not give very useful predictions because there would still be a large amount of unexplained or random variation. With 5 observations a correlation of 0.85 would not achieve statistical significance at the 5% level. For most purposes, even a highly significant regression which accounts for less than about two thirds of the variance is unlikely to be of much practical use.

3.2. Predicting effort using the best subset of measures

To find the best subset, the metric with the largest correlation with effort was identified and then other measures which significantly improved the prediction were added. The metric with the largest correlation with effort was *user-properties* (correlation 0.827). Regression of effort on this alone accounts for about 68% of the variance.

Table 4
Correlations of effort and the size measures

	Effort	Raw-lines	Objects	Named-objects
Raw-lines	0.431			
Objects	0.460	0.757		
Named-objects	0.582	0.858	0.852	
Pages	0.635	0.619	0.696	0.809

Table 5
Correlations among effort and the complexity measures

	Effort	ifs	elses	conditions	whens	propertied-objects	WinEcon-graphs
ifs	**0.411**						
elses	**0.508**	**0.683**					
conditions	0.351	**0.810**	**0.458**				
whens	0.392	**0.813**	**0.470**	**0.976**			
propertied-objects	**0.487**	0.489	0.144	**0.597**	**0.586**		
WinEcon-graphs	− 0.113	0.391	0.356	0.217	0.125	0.174	
steps	0.222	**0.679**	0.370	**0.482**	**0.479**	0.365	0.361

Each of the other measures was added one at a time to the regression. Only the addition of *scrolling-fields* made a significant improvement at the 5% level. Regression on these two measures accounted for about 77% of the variance of effort.

This yields a prediction equation:

$$\text{effort} = 299 + 17.6 \times \text{user} - \text{properties} + 60.2 \times \text{scrolling} - \text{fields}.$$

A plot of residuals against predicted values did not suggest that the regression assumptions are inappropriate here. Adding *forwards* to *user-properties* gave an improvement that was nearly significant at 5%, so it was added to the regression on *user-properties* and *scrolling-fields* but this did not significantly improve the result.

To estimate the accuracy of prediction on new cases, a bootstrap estimate of error was made. The first observation was removed and the regression of effort on *user-properties* and *scrolling-fields* was calculated using the other 24 observations. This regression was then used to predict the effort for the first observation.

This process was repeated by omitting the second observation and then the rest of the observations in turn. This process gives a good estimate of the accuracy of prediction which would be achieved on new data from comparable projects. We hope to identify such projects in the public sector with the expansion of funding for this type of work. The results are shown in Table 6.

The mean squared error is 3316.28 and the standard deviation of the errors is 58.758. This is a potentially usable predictor with a mean magnitude of relative error ($\overline{\text{MRE}}$) of 31.66 and a PRED(0.25) of 0.56 which is just outside Campbell et al. [30] definition of an adequate effort estimation model.. However, although *scrolling-fields* is a portable metric, *user-properties* is ToolBook-specific and, to a degree, project specific which reduces the portability of the effort predictor equation.

Table 6
Bootstrap estimate of predicted effort accuracy

WinEcon chapter	Effort (developer h)	Predicted effort (developer h)	Absolute error(%)
1	350	914.49	161.284
2	672	953.67	41.915
3	700	1003.13	43.304
4	3360	3021.90	10.062
5	1225	910.25	25.694
6	1225	1234.32	0.761
7	2100	1793.01	14.618
8	1890	1531.26	18.981
10	1050	1440.04	37.146
11	1680	1302.62	22.463
12	1470	2242.49	52.550
13	1470	1726.51	17.450
14	1470	1863.11	26.742
15	1400	1299.17	7.202
16	1260	1549.41	22.969
17	1400	674.60	51.814
18	2184	1491.08	31.727
19	2100	2462.63	17.268
20	1225	650.28	46.916
21	420	814.91	94.027
22	3150	2633.65	16.392
23	700	705.43	0.776
24	1050	1225.52	16.716
25	1050	982.62	6.417
26	1050	982.62	6.417

3.3. Predicting effort using the best subset of portable measures

The method used in Section 3.2 was repeated but the analysis is limited to only the group of portable measures. The best of these is *graphics* (correlation with effort 0.707). Regression on this alone accounts for 50% of the variance of effort. Once again, only the *elses* metric gives a significant improvement at the 5% level and *words* is nearly significant. Both of these were added to the regression on *graphics*, and *words* only just fails to reach significance in the presence of both the others. Regression on all three accounts for about 69% of the variance of effort. The effort prediction equation is:

$$\text{effort} = 161 + 10.4 \times \text{graphics} + 5.37 \times \text{elses} - 0.0516 \times \text{words}.$$

Again residual plotting did not suggest any departure from the regression assumptions. Once again a bootstrap estimate of error is obtained by omitting each observation in turn. The results are shown in Table 7.

The mean squared error here is 4558.85 and the standard deviation of the errors is 68.91 with a $\overline{\text{MRE}}$ of 34.46 and a PRED(0.25) of 0.48. This places the resulting effort estimation model in the poor category proposed by Campbell et al.'s [30] criteria. However, this result is usable and it is certainly better than the results obtained by expert estimators predicting development effort [17].

3.4. Size measures

Because of the centrality of the problem of identifying a good size metric for courseware, we looked at the relationships among our four size measures, *raw-lines*, *objects*, *named-objects* and *pages*. Of these, *pages* could be used to estimate learner time, since it was the developers' intention that each page should occupy the students for about 20 min.

Plots show simple linear relationships among our size measures: Fig. 1 showing pages against objects is typical. The matrix of correlations is shown in Table 8: all are significant at the 5% level.

A principle component analysis of the correlation matrix of these four measures showed that the first principle component was almost exactly a sum of the measures (each standardised to a scale with mean zero and standard deviation one) and that this accounted for 82.5% of the total variance. This would suggest the use of a composite size measure in future. However, the number of *pages* could, at least for some projects, be estimated early on whereas the other three measures could only be obtained much later. So *pages* is potentially more useful for estimation, though its correlation of 0.635 with *effort* though highly statistically significant, is too weak to give good predictions by itself.

Table 7
Bootstrap estimate of predicted effort accuracy

WinEcon chapter	Actual effort (developer h)	Predicted effort (developer h)	Absolute error (%)
1	350	618.75	76.785
2	672	1134.64	68.844
3	700	963.05	37.579
4	3360	2785.02	17.113
5	1225	1160.56	5.260
6	1225	1209.14	1.295
7	2100	1484.83	29.294
8	1890	2241.73	18.610
10	1050	1390.13	32.394
11	1680	987.91	41.196
12	1470	2004.90	36.388
13	1470	1379.26	6.173
14	1470	1790.62	21.811
15	1400	1272.93	9.077
16	1260	758.67	39.788
17	1400	1220.86	12.796
18	2184	1471.21	32.637
19	2100	2246.05	6.955
20	1225	606.64	50.479
21	420	971.02	131.194
22	3150	2516.70	20.105
23	700	702.94	0.420
24	1050	2106.54	100.623
25	1050	972.42	7.389
26	1050	1651.51	57.287

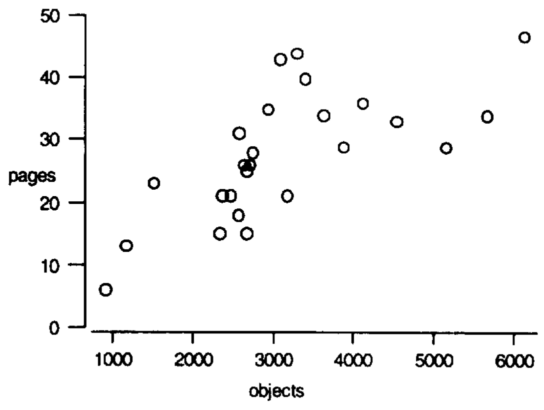

Fig. 1. Pages against objects.

4. Discussion of results

4.1. Winecon

The results of the analysis have provided a starting point for further research into the development of large scale courseware engineering projects in which multiple teams work collaboratively on a number of individual chapters with a common design, programming framework and authoring tool. It is analysed here 'warts and all' with chapters which were re-coded, programmers who left the project, and a major upgrade to the authoring tool during the project. In other words, it is typical of a real courseware development which lasts more than a few months.

4.2. A priori considerations

Disappointingly the a priori considerations did not produce a prediction equation which would give usable results. It is reasonable to assume that size and complexity would have a significant effect on the development effort, but the initial analysis suggests that although this is certainly true, the dependence of effort on these measures is not close enough to produce a useful prediction equation. However, the substantial re-coding required due to the upgrading of ToolBook from version 1.5 to 3 may have affected the results. Also, treating the courseware as one large development may be masking interesting effects and analysis is currently underway to consider the chapters produced by each team individually.

4.3. The best subset of measures

The search for the best predictors of effort resulted in a prediction equation based on user-properties and scrolling-fields. This equation produces useful effort predictions for ToolBook-based courseware. It still leaves the recurring problem associated with most measures research of how to estimate both of these properties at an early stage in the development and what these measures relate to in terms that a project manager will understand. Examination of the role of *user-properties* in the code indicates that they are used to track properties resulting from step cards, graphics and other 'house style' objects. Similarly, *scrolling-fields* are not heavily used throughout the chapters because of style guidelines but when they are used, they tend to link together textual information with examples or simulations. This may give an indirect indication of substantial programming effort to create the example or simulation. This needs further investigation to produce a prediction equation that is easier to use at an early stage in the development process.

Table 8
Correlations matrix

	Pages	Objects	Named objects
Objects	0.696		
Named objects	0.809	0.852	
Rawlines	0.619	0.757	0.858

4.4. The best subset of portable measures

An effort prediction equation which is able to explain 69% of variance, while not good in software measures terms, is certainly better than existing courseware effort estimation methods [28]. The equation makes use of the best set of portable measures which may have widespread use with any courseware authoring system or language. The use of *graphics* in the equation is perhaps not surprising given that it is the dominant media type in this courseware. Excluding the creation of the graphical images which were provided to the development team, the integration of bit mapped images or graphics buttons and their positioning on the page are time-consuming activities. Similarly, analysis of the use of *else* in the code indicates the existence of complex student interaction or decision making within the code. In ToolBook the *else* condition appears only in conjunction with *if* or *when*, neither of which appeared to be significant. Analysis of the source code indicated that the *if* or *when* without the *else* condition appeared to be used for simple interactions or code decisions. The *else* condition was mainly used in complex student interactions or code decisions which would require considerable programming effort. In the presence of *graphics* and *elses*, the effect of *word* is to reduce the predicted effort. This could be because the presence of a lot of text, though involving the programmer in the effort of typing it in, is associated with a simpler programming structure for the chapter.

4.5. Size

Our size measures are all closely related, and one of them could be related easily to the 'learner time' favoured in the industry, but more work needs to be done, especially on specifying more precisely what 'learner time' means.

5. Conclusion

The dominance of media and the user interface in the final product makes courseware significantly different from other software so that most existing traditional software measures research is of limited use. With the growing importance of courseware and multimedia in education and training it is essential that research is done to gain a better understanding of the development process with a view to improving productivity, quantifying and improving product quality and effort estimation. This study was of a project where several teams worked as a consortium on a single large scale courseware development. The results clearly indicate that there are measures based on code features which can contribute to improved ability to estimate development effort. We can suggest ways in which these code features are related to attributes which could be identified from the specification and the next major task is to try to relate directly the development effort to these proposed early specification measures.

References

[1] P. Davidsen, J.J. Gonzalez, D.J. Muraida, J.M. Spector, J.M. Tennyson, Applying system dynamics to courseware development, Computers in Human Behavior 11 (2) (1995) 325–339.

[2] R. Canale, S. Wills, Producing professional multimedia: Project management issues, British Journal of Educational Technology 26 (2) (1995) 84–93.

[3] J. Baker, One man and his dog, Interact 1 (3) (1994) 16–17.

[4] I.M. Marshall, W.B. Samson, P.I. Dugard, W.A. Scott, Predicting the development effort of multimedia courseware, Information and Software Technology 36 (5) (1994) 251–258.

[5] J. Verner, G. Tate, A software size model, IEEE Transactions on Software Engineering 18 (4) (1992) 265–278.

[6] I.M. Marshall, W.B. Samson, P.I. Dugard, G.R. Lund, The mythical development to learner time ratio, Computers and Education 25 (3) (1995) 113–122.

[7] J. Jay, K. Bernstein, S. Gunderson, Estimating computer-based training development times, Research Institute for Behavioural and Social Sciences, Alexandria, VA, 1987.

[8] B.W. Boehm, Software Engineering Economics, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[9] D.R. Jeffery, C.G. Low, M. Barnes, A comparison of function point counting techniques, IEEE Transactions on Software Engineering 19 (5) (1993) 529–532.

[10] B. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy, R. Selby, Cost models for future software life cycle processes: COCOMO 2.0, University of Southern California, Los Angeles, CA, 1994, pp. 1–31.

[11] R.D. Banker, R.J. Kauffman, R. Kumar, An empirical test of object-based output measurement metrics in a computer aided software engineering (CASE) environment, Journal of Management Information Systems 8 (3) (1992) 127–150.

[12] S. Price, L. Cheah, P. Hobbs, Software quality in end-user programming and object based rapid application development (RAD), in: BCS Software Quality Management Conference, British Computer Society, Cambridge, 1996.

[13] M.J. Bunzel, S.K. Morris, Multimedia Application Development: Using DVI Technology, McGraw-Hill, New York, 1992.

[14] L.A., Laranjeira, Software size estimation of object-oriented systems, IEEE Transactions on Software Engineering 16 (5) (1990) 510–522.

[15] C.R. Finnie, G.E. Wittig, Effect of system and team size on 4GL software development productivity, South African Computer Journal 11 (1994) 18–25.

[16] P. Fairweather, A. O'Neal, The impact of advanced authoring systems on CAI productivity, Journal of Computer-Based Instruction 11 (1984) 90–94.

[17] I.M. Marshall, Evaluating courseware development effort estimation measures and methods, University of Abertay Dundee, Dundee, 1996.

[18] G. Senbetta, An inquiry of time and cost estimating for computer-based training courseware design and development as determined by modified Delphi method, Purdue University, 1991.

[19] B. Lisewski, C. Settle, Teaching with multimedia: A case study in weed biology, Active Learning 3 (1995) 28–35.

[20] J. Orlansky, J. String, Cost-effectiveness of computer-based instruction in military training, Institute for Defence Analysis, 1979.

[21] L. Briand, S. Morasca, V.R. Basili, Defining and validating high-level design metrics, University of Maryland College Park, MD, 1996.

[22] HEFC, TLTP – Phase I, Higher Education Funding Council, 1996.

[23] HEFC, TLTP – Phase II, Higher Education Funding Council, 1992.

[24] J. Sloman, WinEcon software review, Economic Journal 5 (429) (1995).

[25] S. Price, P. Hobbs, ToolBook 1.5., The Economic Journal 4 (425) (1994).

[26] S. Price, P. Hobbs, Asymetric ToolBook 3.0 – Implications for developers, Centre for Computing in Economics, University of Bristol, Bristol, 1995.

[27] R.E. Courtney, D.A. Gustafson, Shotgun correlations in software measurements, Software Engineering Journal (1993) 5–13.

[28] I.M., Marshall, W.B. Samson, P.I. Dugard, Multimedia courseware, Never mind the quality, how much will it cost to develop?, Association for Learning Technology Journal 3 (1) (1995) 110–117.

[29] E.I. Oviedo, Control flow, data flow and program complexity, in: M. Shepperd (Ed.), Software Engineering Metrics, Vol. 1: Measures and Validations, McGraw-Hill Book Company, London, 1993, pp. 52–65.

[30] R.L. Campbell, S.D. Conte, M.K. Rathi, Early prediction of software size and effort, Purdue University, 1988.