Evaluating Courseware Development Effort Estimation Measures and Models

by

Ian Mitchell Marshall BSc (Hons), BA, Dip Ed Tech

A thesis submitted in partial fulfilment of the requirements of the University of Abertay Dundee for the degree of Doctor of Philosophy

7 November 1996

I certify that this thesis is the true and accurate version of the thesis approved by the examiners

Signe[redacted] Date 22/4/96

(Director of Studies)

TABLE OF CONTENTS

LIST OF TABLES

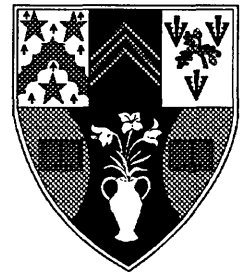
LIST OF ILLUSTRATIONS

ACKNOWLEDGEMENTS

Recognition and grateful appreciation is expressed to all the courseware developers who provided development data and expert opinion on *effort* estimation. Special thanks are extended to Mr S Price and Mr P Hobbs of the Centre for Computing in the Social Sciences based at Bristol University who allowed me unlimited access to WinEcon documentation, development records and source code. I am also grateful to the numerous teams of courseware developers who provided data. I would also like to thank Ms P McDonald and Mr N Tregham whom I employed for short periods as research assistants to do the "boring bits" such as collecting and entering questionnaire results or creating courseware and measuring *effort* expended respectively.

My gratitude is extended to the School of Informatics Computing Research Advisory Group who funded a short sabbatical to write up the work and to all the staff who taught my classes during my period of absence. Special recognition and thanks are due to my supervisors Dr WB Samson and Ms P Dugard for all their help, advice and encouragement during the work and writing up of this thesis. Last but not least I would like to thank my wife Patricia for her patience in putting up with me for the last few years and for her perseverance in proof-reading yet more stuff on "cost drivers".

PERMISSION TO COPY



UNIVERSITY

of

ABERTAY DUNDEE

Author : Ian Mitchell Marshall

Title : Evaluating Courseware Development Effort Estimation Measures and Models

Qualification : Doctor of Philosophy

Date of submission : 08 November 1996

I agree that a copy of this may be made in whole or in part of the above-mentioned thesis without further reference to the undersigned. This permission covers only single copies for study purposes and is subject to the normal conditions of acknowledgement.

Signature [redacted]

School of Informatics
University of Abertay Dundee
Bell Street, Dundee DD1 1HG
[redacted]

ABSTRACT

The aim of the work described in this thesis is to establish a theoretical and practical framework for evaluating courseware effort estimation models. Existing research into this area is thoroughly reviewed. Expert estimation, algorithmic and non-algorithmic estimation methods are evaluated in terms of well defined criteria adapted from software metrics research. Developers' opinion on courseware effort estimation and analysis of the significant factors in improving the estimation accuracy will be discussed. A rigorous measurement paradigm to describe and assist in data collection for courseware effort estimation is defined.

The results from twelve studies in to courseware effort estimation are used to identify seventy seven productivity adjustment factors. The productivity adjustment factors are classified into six broad groupings. The rating scale used in existing effort estimation models are presented. Nineteen critical productivity adjustment factors which affect courseware effort estimation methods are also identified.

The implementation of the measurement paradigm and productivity adjustment factors are evaluated using courseware development data in order to prepare effort estimation models. Four courseware development case studies are used to evaluate the measurement paradigm for standardising courseware development effort measures and data collection. In addition, effort estimation models are developed using courseware development data from existing studies and a large courseware development.

Analysis of the effort estimation models indicated that significant relationships existed between effort, learner time, key productivity adjustment factors and code measures. The effort estimation models produced disappointing results when compared against established criteria for evaluating software effort estimation models. However, despite this a number of important points were highlighted which will result in the development of better models in the future. Finally, the results and methodology used in this thesis are critically evaluated before further work and future directions for research are explored.

ABBREVIATIONS, SYMBOLS AND NOTATIONS

Abbreviation	Term	Description
Lh	*Learner hours*	Learner time
Lh Dh^{-1}	*Learner hours per Developer hour*	Productivity
Dh Lh^{-1}	*Developer hours per Learner hour*	Effort to learner time ratio
Dh	*Developer hours*	Effort
D	*Elapsed days*	Elapsed time
H	*Hours*	Development time
£ or $	*Currency (Pound or Dollars)*	Cost
MRE	*Mean Magnitude of Relative Error*	Predictive accuracy of model
Pred(l)	*Prediction at level l*	Predictive accuracy of model
£ or $ Dh^{-1}	*Currency per Developer hour*	Developer effort charge

1. INTRODUCTION

> "...the multimedia education market has the potential to be a bigger grossing industry than the entire movie business. If we get a domestic market for multimedia education resources into play and make it work. ... we can then seriously turn our attention to taking advantage of export opportunities, because we have the English language, we are recognised as a source of knowledge and have a strong tradition in teaching backed up by powerful accreditation systems." (Puttnam 1996)

Sir David Puttnam's (1996) presentation to the University Parliamentary Group highlighted the strengths and the weaknesses of the United Kingdom as a developer and supplier of multimedia courseware. This presentation extended Baker's (1994) theme in which he humorously described the failure of teachers, companies, organisations and governments over the last 25 years to deliver the volume of quality courseware which would spark the active learning revolution.

While the cost of hardware required to develop and deliver multimedia courseware has fallen, the real cost of developing quality courseware remains high as does the risk of failure associated with its production (Jay et al. 1987; Hobbs and Price 1994; Soloman 1994; Tennyson et al. 1995). Multimedia courseware developers need to be able to accurately estimate the development effort required to undertake a project. Without the ability to estimate effort accurately every courseware development project becomes a risky gamble in which the developer commits precious resources on the hope of delivering effective courseware on time and within budget.

In this Chapter the problem area will be defined along with the overall structure of this thesis.

1.1 DEFINING THE PROBLEM AREA

When Gery (1986) asked rhetorically "How long does it take to develop an hour of [courseware]?" she did not expect her response "How many angels can dance on the head of a pin?" to sum up the extent of published knowledge on courseware effort estimation almost 10 years later. Considering the economic significance of multimedia courseware development, the lack of published research into effort estimation is discouraging. It would not be so disheartening if multimedia courseware developments were accurately estimated, planned and implemented but there is unfortunately some evidence to indicate that this is not the case (Jay et al. 1987; Canale and Wills 1995; Tennyson et al. 1995). Projects are often subject to time and cost overruns or can even come to a standstill with the risk of delivering less than was promised or nothing at all (Knight 1992; Whitten 1992).

Some of the risks associated with multimedia courseware development may be reduced if realistic estimates were used to plan projects. While there are a number of published courseware effort estimation methods and some commercial organisations claim that they use rigorous methods to derive their estimates, there has been no systematic attempt to research courseware development effort estimation models.

1.1.1 Aim of this project

The aim of this project is to systematically investigate and evaluate courseware effort estimation methods, measures and models. It forms part of a body of work leading to the development of a new discipline or approach called courseware engineering (De Diana and van Schaik 1993). Goodyear (1995) described courseware engineering as:

> "... an emerging set of practices, tools and methodologies which result from attempts to take an engineering approach to the production of courseware. This engineering approach is in contrast to the craft or artisan approach. The engineering approach emphasises the use of principled methods rather than intuition. It values replicability of process and results rather than idiosyncratic creativity. Its products are complex and need multi-disciplinary teams for their creation. ... They need to be managed and to know when they are achieving or failing to achieve appropriate standards in the production process as well as the product."

This mirrors the calls which led to the development of a discipline called "software engineering". Within software engineering, the development of software metrics has resulted in a better understanding of the "... entities of interest in software development in terms of products, processes and resources" (Fenton 1991). It is only by applying the rigour of measurement to courseware that we can hope to manage the production process or know when its development is surpassing or failing to achieve an appropriate standard (Goodyear 1995).

The problem is where to start developing a better understanding of the process and the product of courseware engineering through the application of metrics techniques and measurement theory. De Diana and van Schaik (1993) argue that "In most educational software development projects the effectiveness of the final product and the efficiency of development work are of paramount importance, if not the major goals". While the effectiveness issue will be partially covered in this thesis, the main aim is to develop a paradigm to measure the "efficiency of the development work". This forms the basis for a better understanding of the "efficiency of the development work" and therefore to the accurate prediction of multimedia courseware development effort.

1.1.2 Research objectives

This thesis presents the results of research which aimed to achieve the following objectives.

- Evaluate existing courseware effort estimation methods against well defined model criteria
- Synthesise and evaluate an original measurement paradigm to standardise courseware effort estimation data collection and evaluation
- Synthesise and evaluate productivity adjustment factors which affect courseware effort estimation using an original classification scheme
- Synthesise and evaluate courseware effort estimation models using courseware development data

1.1.3 Research methodology

To achieve the research objectives a substantial literature review was carried out to identify existing courseware effort estimation methods and associated literature. Based on this, an evaluation of existing effort estimation methods was used to prepare a measurement paradigm and identify productivity adjustment factors. Courseware development data from existing studies, published and unpublished developments and a detailed analysis of a large scale courseware development was undertaken to evaluate the proposed framework. A small scale survey into courseware developers' opinions on effort estimation was also carried out to establish a baseline for modern courseware development. The results of this analysis will be presented as a series of individual case studies.

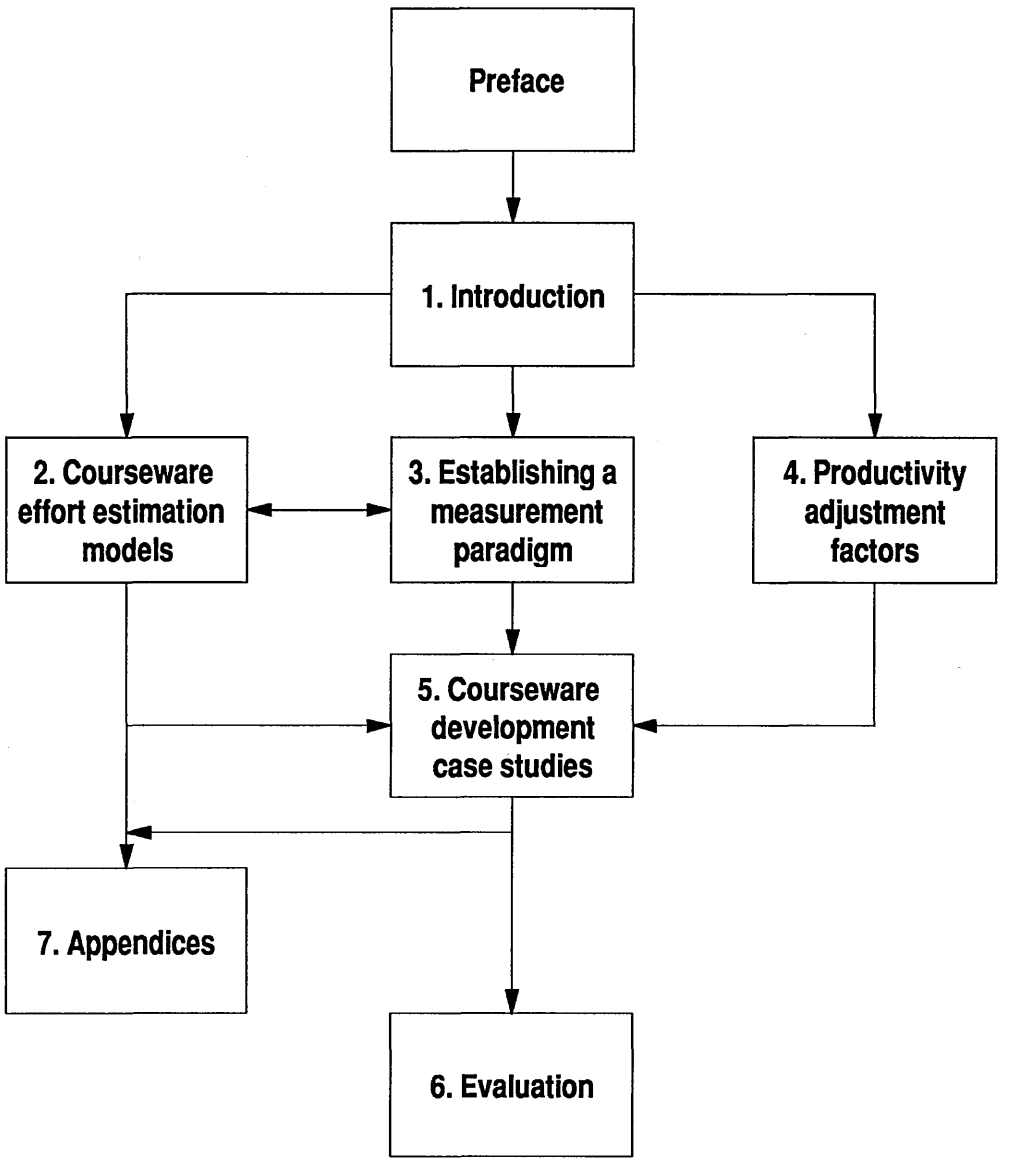
1.2 OVERVIEW OF THESIS STRUCTURE

This Section presents an overview of the structure of the thesis on a Chapter by Chapter basis.

1.2.1 Structure diagram

Each Chapter is relatively self contained with its own references and endnotes. However, as Figure 1.1 shows diagramatically, each Chapter is related directly or indirectly to at least one subsequent Chapter.

Figure 1.1 - Overview of thesis structure



1.2.2 Chapter descriptions

Preface

The Preface contains the basic information about the thesis including contents, lists of tables and illustrations, acknowledgements, permission to copy, abstract and abbreviations, symbols and notations.

Chapter 1 - Introduction

The Introduction presents an overview of the problem area, thesis aims, research objectives and overview of the thesis structure.

Chapter 2 - Courseware effort estimation models

This Chapter presents a summarised version of the measurement paradigm defined in order to assist discussion of existing courseware effort estimation methods. Research in software metrics is used to establish criteria for evaluating courseware effort estimation methods and models. Existing effort estimation methods are classified into three broad groupings; expert opinion, non-algorithmic, and algorithmic methods. The limitations of the existing methods are then evaluated using Boehm and Wolverton's (1980) criteria for software effort estimation models.

Chapter 3 - Establishing a measurement paradigm

Each new courseware authoring tool or development system announces its arrival with a claim for increased productivity along with improved ease of use. In less than a decade there has been at least an eight-fold improvement in productivity reported in the literature (Merrill et al. 1990; Marshall et al. 1994; Orey et al. 1994). At the same time, the range of media, visual quality and complexity of the finished product has increased. Is this increase in productivity real or does it reflect variability in the product or the development process or is it a result of inconsistent definitions used to measure effort? A case study will be used to present the results of a small sample survey of courseware developers' opinions on courseware effort estimation. This will show that the only significant factor in improving the accuracy of the effort estimation is the use of a rigorous method. It can be argued that by carefully defining terms, development effort estimation problems could be solved. This hypothesis will be explored by defining a rigorous measurement paradigm to describe the measures required to collect data for

multimedia courseware effort estimation methods. The measurement paradigm presented here will then be evaluated in Chapter 5.

Chapter 4 -Productivity adjustment factors

In this Chapter seventy seven productivity adjustment factors are identified from published studies into courseware effort estimation methods. The productivity adjustment factors will be classified into six broad headings which will reduce the overlap in productivity adjustment factors found in existing effort estimation methods. Rating scales for each of the existing productivity adjustment factors are also presented. The highest ranked productivity adjustment factors are identified for further analysis in a case study presented in Chapter 5.

Chapter 5 - Courseware development case studies

Four case studies in courseware effort estimation is discussed in Chapter 5. These case studies establish the need for a more rigorous method of estimating courseware effort and also evaluate the use of the measurement paradigm and productivity adjustment factors presented earlier. Data from two existing courseware developments are re-analysed to establish the relationships missed or not considered by the original authors. A case study based on analysis of a large scale courseware development is also be discussed.

Chapter 6 - Evaluation

In this Chapter the original contribution to knowledge is highlighted and critically evaluated. Strengths and weaknesses in the thesis are discussed before future directions for research are presented.

Chapter 7 - Appendices

Chapter 7 contains "rules of thumb" for courseware effort estimation which were collected during the research for this thesis. In addition, summaries of the courseware effort estimation data will be presented.

1.3 SUMMARY

In this Chapter the problem of accurately estimating development effort was defined in terms of the need to reduce the risk associated with producing courseware and to develop a better understanding of the process and product as part of a new discipline called courseware engineering. An overview of the thesis aims, research objectives and the methodology to be used to achieve these objectives was discussed along with an outline of the thesis structure.

In the next Chapter a comprehensive review of existing literature related to courseware effort estimation is presented.

1.4 REFERENCES

Baker, J., 1994. One man and his dog. *Interact.* **1(3)**: pp. 16-17.

Boehm, B.W. and Wolverton, R.W., 1980. Software cost modelling: Some lessons learned. *Journal of Systems and Software.* **1(3)**: pp. 195-201.

Canale, R. and Wills, S., 1995. Producing professional multimedia: project management issues. *British Journal of Educational Technology.* **26(2)**: pp. 84-93.

De Diana, I. and van Schaik, P., 1993. Courseware engineering outlined: An overview of some research issues. *Education & Training Technology International.* **30(3)**: pp. 191-211.

Fenton, N., Editor 1991. *Software metrics: a rigorous approach.* London: Chapman-Hall.

Gery, G., 1986. How long does it really take? Estimating CBT development cost. *DATA Training.* **5(8)**: pp. 31-36.

Goodyear, P., 1995. Infrastructure for courseware engineering. In: R.D. Tennyson and A.E. Barron, Editors *Automating instructional design: computer-based development and delivery tools.* Berlin: Springer-Verlag, pp. 11-31.

Hobbs, P. and Price, S., 1994. Educational project management; Repeat after me. In: *ALTC 94, Enabling Active Learning,, University of Hull, Hull, UK, 18-21 September 1994.*

Jay, J., Bernstein, K. and Gunderson, S., 1987. *Estimating computer-based training development times.* ARI Technical Report (765), October 1987, Alexandria, VA: Research Institute for Behavioural and Social Sciences.

Knight, P., 1992. Factors to consider in evaluating multimedia platforms for widespread curricular adoption. *Educational Technology.* **May**: pp. 25-27.

Marshall, I.M., Samson, W.B. and Dugard, P.I., 1994. Multimedia courseware. Never mind the quality, how much will it cost to develop? In: *Association for Learning Technology 94, University of Hull, Hull, 18-21 September 1994.*

Merrill, M.D., Li, Z. and Jones, M.K., 1990. Limitations of first generation instructional design. *Educational Technology.* **30(1)**: pp. 7-11.

Orey, M. et al., 1994. Development efficiency and effectiveness of alternative platforms for intelligent tutoring for the mobile subscriber radio-telephone terminal. *Computers and Education.* **22(4)**: pp. 301-313.

Puttnam, D., 1996. *Presentation to Parliamentary University Group.* 27 February 1996, 16 Great College Street, London SW1P 3RX: Parliamentary University Group.

Soloman, M.B., 1994. What's wrong with multimedia in higher education? *Technological Horizons in Education Journal.* **21(7)**: pp. 81-83.

Tennyson, R.D. et al., 1995. Employment of system dynamics in modelling of instructional design (ISD). In: R.D. Tennyson and A.E. Barron, Editors *Automating instructional design: computer-based development and delivery tools.* Berlin: Springer-Verlag, pp. 603-609.

Whitten, W.B., 1992. The hurdles of technology transfer. *Educational Technology.* **May**: pp. 19-24.

2. COURSEWARE EFFORT ESTIMATION METHODS

> "How is the planning and implementation of computer-based learning environments being controlled and managed? This question goes to the heart of current efforts to make effective use of advanced learning technologies. The answer, unfortunately, is that there is no empirically established methodology for controlling and managing large-scale courseware development efforts. As a consequence, many courseware development projects end behind schedule, over budget, and with only marginally effective learning environments." (Tennyson et al. 1995)

Multimedia has been used in education and training for over fifteen years to deliver instruction using computers to control interactive video and other media (Stewart and Bryce 1981). From these primitive beginnings to the development of commercially available multimedia computers controlling CD-ROM or interactive video the major constraint on widespread use has not been the hardware but rather the development of courseware. The availability of low cost multimedia production and delivery systems (Stack 1990; Barron and Fisher 1993; Baker 1994) has resulted in an increase in interest in the use of learning materials based on this technology. However, despite the reduction in cost of hardware and improved functionality of authoring software, the development effort required to produce multimedia courseware is still substantial.

Soloman (1994) in an article on the problems of using and developing courseware stated that successful production of multimedia requires hundreds of developers hours planning, developing and revising. Making this level of commitment to the development of multimedia courseware requires the manager of the project to make accurate estimates of the effort required before the project begins. To do this, it is essential to have a good understanding of the multimedia courseware development process and effort estimation.

In this Chapter, literature on courseware effort estimation methods will be discussed under the following Section headings:

- Rationalising the tower of Babel
- Courseware effort estimation methods
- Evaluation of existing methods
- Summary

First a consistent terminology which will be used throughout this thesis must be established.

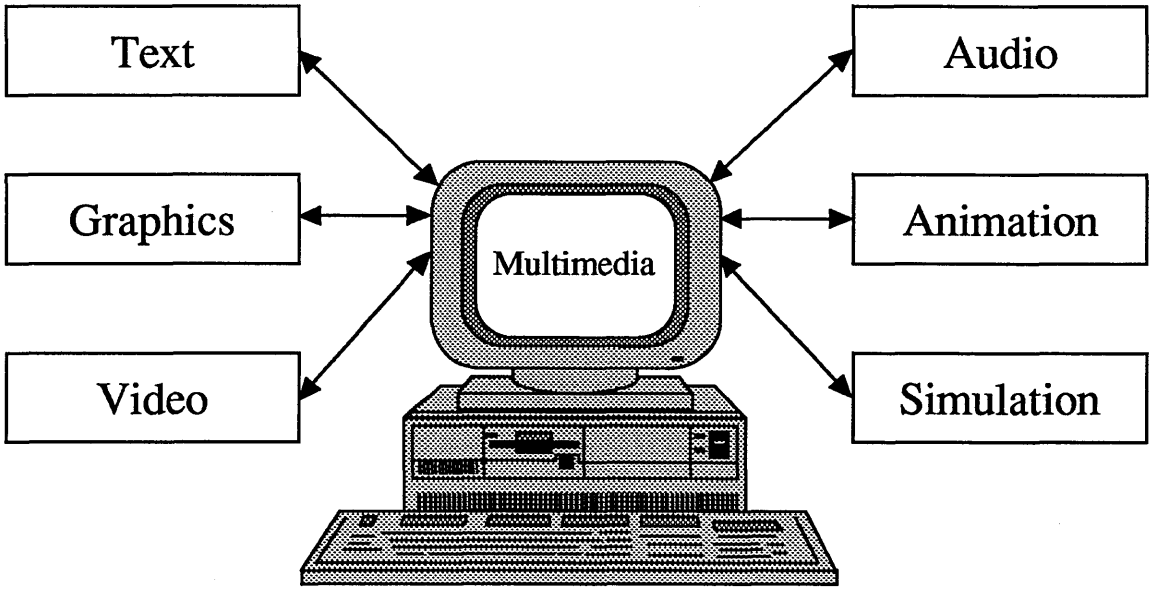
2.1 RATIONALISING THE TOWER OF BABEL

One of the problems identified by Marshall et al. (1995) is the confusing array of terms used to describe multimedia courseware product and process measures. Even more confusing is where despite there being agreement on what is being measured, a diverse range of terms is used to describe the same measures. It is very difficult to draw meaningful conclusions if every time a term is used it has to be defined and compared to the related measures and terms. As de Diana and Schaik (1993) highlighted in relation to courseware engineering, there is a need to establish a consistent terminology in an emerging discipline. In this Section, the terminology that will be used throughout this thesis will be defined.

2.1.1 What is multimedia?

This is a difficult question to answer concisely because the literature abounds with definitions of the term. Fox (1991) stated that "... to qualify for the title multimedia, an application needs only to incorporate two or more of the following: still or animated graphics, still or motion video, audio or text and numerical data." Others claim that even this relatively recent definition does not encompass the rapidly evolving range of technologies which are now described as multimedia (Gayeski 1992). At the other end of the spectrum there are those who claim that multimedia is nothing new and that media, whether computer controlled or not, is always at least plural (Ralston 1991). However, Galbreath's (1992) definition of multimedia as "... the combination of two or more communication media under computer control" as shown in Figure 2.1 will be used in this thesis.

Figure 2.1 - Diagrammatic representation of multimedia



2.1.2 What is courseware?

Gery (1987) described computer-based training as an "... interactive learning experience between a learner and a computer in which the computer provides the majority of the stimulus, the learner must respond and the computer analyses the response and provides feedback to the learner." Variations on this description can be found in the literature attached to a bewildering array of three letter acronyms. Table 2.1 shows an extended version of Gery's (1987) "Chinese Menu" which explains most of these three letter acronyms.

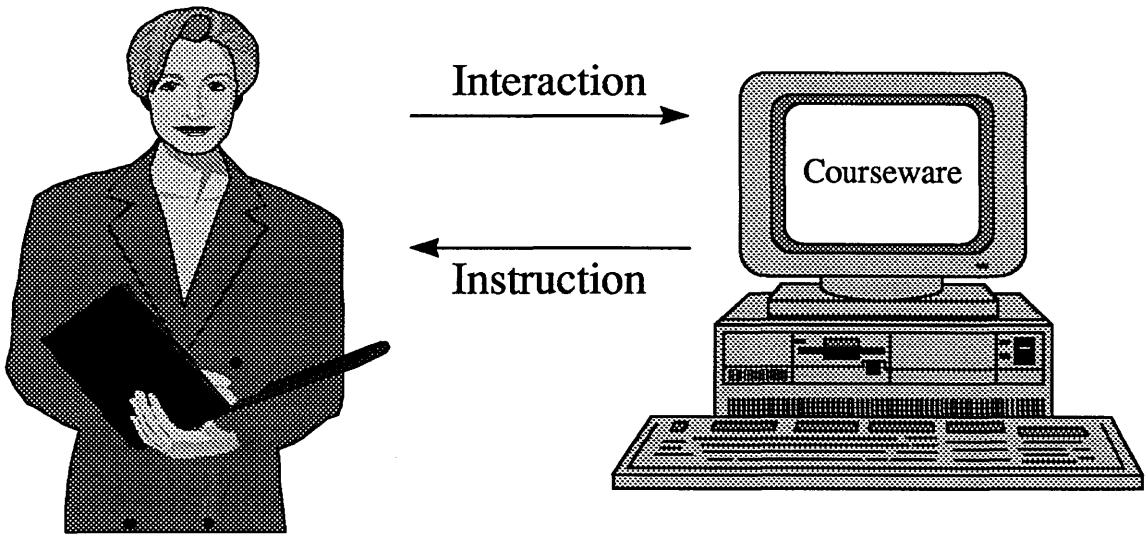
Table 2.1 - 'Chinese Menu' (Gery 1987)

Pick your own terminology by selecting one 'term' from each column		
A	**B**	**C**
Computer	**A**ssisted	**I**nstruction
Multimedia	**A**ided	**L**earning
	Managed	**E**ducation
	Based	**T**raining
	Enhanced	**T**eaching
	Mediated	**D**evelopment
	Interactive	**S**tudy
		Courseware

In this thesis, courseware is the generic term used to describe software, media and support material designed with an educational or instructional purpose (Goodyear

1995). Figure 2.2 shows a representation of an instruction interaction mediated by courseware.

Figure 2.2 - Diagrammatic representation of courseware



2.1.3 What is being measured?

Mikos et al. (1987b) in a series of related studies asked courseware development experts to estimate the "time" required to develop a number of different training scenarios. This highlights one of the problems found in the literature. Authors tend to use *development time*, *effort*, *cost* and *effort to learner time ratios* interchangeably, and while they are related, they are not the same. Marshall et al. (1995) proposed a paradigm to measure various aspects of multimedia courseware development. The following informal descriptions of the paradigm metrics and their associated units will be used consistently to discuss the literature on multimedia courseware effort estimation before a formal definition is presented in Chapter 3. In direct quotations the paradigm terms will be substituted using the normal convention of square brackets [].

Elapsed time

As the name suggests, *elapsed time* measures the total time in *days (D)* from the start to the end of the project. It is not commonly found in courseware development literature.

Development time

Development time is the basic measure of the number of hours taken to develop the

multimedia. During this period *development time* is measured in *hours (H)* elapsed during the multimedia courseware development. It excludes non-working time such as weekends, holidays and lunch breaks.

Effort

In the literature effort is seldom used as a measure in courseware estimation although it is the normal measure in software engineering. Effort provides a measure of the total amount of productive work expended developing the courseware and is measured in *developer-hours (Dh).*

Cost

Cost describes in local currency the monetary value of the effort and other costs associated with producing the courseware.

Learner time

In the literature the amount of time spent by the learner using the courseware is known by a diverse range of names. To prevent confusion this will be described consistently as *learner time* and is measured in *learner-hours (Lh).*

Productivity

Symons (1993) defined a range of productivity measures for software size estimation. The basic measure of *productivity* is given by dividing the output by the input. The normal output measure is *learner time* the courseware delivers and the input is the development effort. *Productivity* is measured in units of *mean-learner-hours-per-developer-hour* ($\overline{Lh}\, Dh^{-1}$).

Effort to learner time ratios

The widely used "development to delivery time ratio" was originally measured in terms of "average man-hours expended per CBI lesson hour" (Hurlock and Slough 1976). However, authors have recently dropped the effort element in favour of the less descriptive "hours of development for each hour of instruction" (Orey et al. 1994). To reduce ambiguity, effort *to learner time ratio* will be used consistently and measured in *developer-hours-per-mean-learner hour* ($Dh\, \overline{Lh^{-1}}$). This makes clear its inverse relationship with productivity.

2.1.4 How accurate is it?

In common with most new disciplines there is no established measurement or statistical evaluation paradigm to follow. To reduce ambiguity the following terms and techniques from software metrics will be used to evaluate accuracy in prediction or reliability of courseware effort estimation methods and models.

Mean magnitude of relative error

Conte Shen and Dunsmore (1986) proposed the use of the mean magnitude of relative error ($\overline{MRE}$) to evaluate the predictive accuracy of estimation models. The $\overline{MRE}$ is calculated using the following equation.

$$\overline{MRE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{E_{actual} - E_{estimate}}{E_{actual}}\right|$$

where E_{actual} is the actual effort and $E_{estimate}$ is the result produced by the estimation method. The smaller the $\overline{MRE}$ value the better the set of predictions produced by the estimation model. However, a estimation method which produces a small $\overline{MRE}$ may contain a number of predictions that are poor.

Prediction at level *l*

To overcome the $\overline{MRE}$ limitation a second measure is normally used to measure the proportion of the estimates with $MRE \leq l$. The prediction at level l or $PRED(l)$ is calculated using the following equation.

$$PRED(l) = \frac{k}{n}$$

where k is the number of projects in the sample with $MRE \leq l$ and n is the total number of projects in the sample. The values of $PRED(l)$ can vary between zero and one with the larger value indicating a better estimation method.

Evaluating an estimation method

Using a combination of $\overline{MRE}$ and $PRED(l)$ provides a convenient criterion to evaluate the predictive accuracy of an estimation method. Conte Shen and Dunsmore (1986)

proposed that the acceptable criterion for a software effort prediction model is a $\overline{MRE} \leq 0.25$ and $PRED(0.25) \geq 0.75$. This criterion has been extended by Campbell, Conte and Rathi (1988) to provide the evaluation scheme shown in Table 2.2.

Table 2.2 - Effort estimation evaluation scheme (Campbell et al. 1988)

Method rating	Estimation method evaluation criteria		
Good	$\overline{MRE} \leq 0.25$	AND	$PRED(0.25) \geq 0.75$
Acceptable	$\overline{MRE} \leq 0.30$	AND	$PRED(0.30) \geq 0.70$
Poor	$\overline{MRE} > 0.30$	OR	$PRED(0.30) < 0.70$

In software size and effort estimation this scheme is arbitrary but widely accepted (Lokan 1996). There is no equivalent scheme for courseware effort estimation. The only reference found was Mikos et al. (1987b) who proposed that a courseware estimate which was within 25% of actual effort[1] is accurate. They also felt that a more realistic target was to aim for estimates within 20% of actual effort. This does not contradict the software estimation evaluation scheme. To ensure compatibility with software effort estimation standards, the evaluation scheme shown in Table 2.2 will be used.

2.2 COURSEWARE EFFORT ESTIMATION METHODS

Multimedia courseware can be developed in any number of alternative ways (Goodyear 1995), therefore selecting the most efficient option is very important since paying for development effort is expensive (Hannafin and Peck 1988). The obvious question is how can a project manager ensure the development is efficient and effective? Confronted with a number of alternative courseware development methods how does a project manager select the most efficient option or even measure the consequences of that selection?

In software engineering, the study and use of metrics is evolving from an informal to a more formal paradigm (Fenton 1991; Fenton 1994; Kitchenham et al. 1995). Can the same be said for courseware metrics? Well, it would appear not given that in the first Section of this Chapter a consistent terminology had to be defined. Courseware metrics is still in the informal stage and part of this research is to establish a more formal method for at least multimedia courseware effort estimation. In the following Section existing courseware estimation methods will be described.

2.2.1 Existing estimation methods

A review of the literature found references to fifteen methods used to estimate multimedia courseware development effort. Other methods may exist; Golas (1993) made reference to "...automated cost-estimation tools developed to serve specific purposes" which had been developed by a number of organisations. Jay, Bernstein and Gunderson (1987) referred to three other methods which were in development in 1987 but, unfortunately, no other reference could be found to these methods.

The fifteen methods found have been classified under the following three broad headings:

- Expert opinion
- Non-algorithmic methods
- Algorithmic methods

These broad classifications will be used to describe the methods identified in the literature. All the methods described have been converted to use the paradigm measures

and units in Section 2.1. A brief description of the original terminology and units used by each method can be found in the Endnotes at the end of this Chapter.

2.2.2 Expert opinion

A small informal survey of multimedia project managers indicated that up to 93% of the respondents only used expert opinion to prepare estimates (Marshall et al. 1994b). The various methods described by the project managers and found in the literature matched Boehm's (1981) descriptions of "Educated Guessing", "Expert Judgement", "Pricing To Win" or "Market Rate". In these methods the estimator constructs the estimates using knowledge and experience of:

- previous projects
- what the customer can afford
- staff availability

While some developers said they prepared the entire estimate "in their head" others formally decomposed the project into phases and tasks which they then estimated individually.

Whatever the method is called, courseware developers do not seem to have the ability to accurately estimate courseware development *effort* or *cost*. In their survey of one hundred and seventy nine courseware developers, Jay, Bernstein and Gunderson (1987) reported that only 11% of developers claimed to be able to estimate within 5% of the actual development *cost*[2]. They also found that experienced developers were no better than inexperienced developers in estimating courseware development *cost*. However, inexperienced developers were more likely to produce estimates with absolute errors greater that 20% of the actual *costs*. Using a modified Delphi technique Senbetta (1991) asked twenty five experts to estimate courseware *effort to learner time ratios*[3] for a well-defined courseware specification. The initial round produced estimates of *effort to learner time ratios* in which the highest value was 27.5 times greater than the lowest value for the same specification and *learner time*. Second round estimates, although improved, still produced a range in which the highest value was 6.5 times greater than the lowest value.

These wide variations in estimates may reflect the expert's personal experience or even incomplete data on which to base the judgement. Whatever the cause it is unlikely that educated guessing forms the basis for a reliable and verifiable method of effort estimation without a database of historical courseware data on which to base the estimates. However, O'Neil (1987) reported that he "... is not aware of organisations which have reliably collected data on [courseware] development". Furthermore, he asserted that "... most [courseware] *cost* estimates are not derived systematically; they are based on educated guesses or intuition". Even when organisations collect data there is a reluctance to use this in estimation due to "unique" factors or missing information (Jay et al. 1987).

In the remainder of this Section estimation methods which try to improve on educated guessing will be described under the following headings:

- Rules of thumb
- Work breakdown
- Work breakdown with project complexity adjustment
- Scene counting
- Industry averages

The methods described are based on published accounts of methods used to estimate the development effort required to produce courseware. Other methods and variations on the methods described exist. The aim here is to provide a basis for comparison of expert opinion against the more formal methods presented later.

Rules of thumb

Parker (1983) described rules of thumb as "... a way to impose meaning on piles of information; they are little Mae Wests that keep one afloat in a sea of cold data". Simply stated a rule of thumb is a general guideline that an expert develops to make quick decisions about courseware developments without having to carry out detailed analysis (Lee and Zemke 1987). Lee and Zemke (1987) also described rules of thumb as easy to remember guidelines developed from expert experience and, while they are less precise than a mathematical formula, they tend to be more useful than a straight guess.

A number of authors have explored the use of rules of thumb used by experts in courseware effort estimation (Lee and Zemke 1987; Mikos et al. 1987a; Mikos et al. 1987b; Casey et al. 1988; Mikos et al. 1988). However, these studies tended to collect rules of thumb on a haphazard basis. They depended on pronouncements from the Oracle rather than undertaking a serious attempt at expert knowledge elucidation. This is perhaps indicated by the fact that Mikos et al. (1987b; 1988) were surprised to find that experienced participants in their estimation exercises were unfamiliar with the concept of rules of thumb and had difficulty generating any to help in courseware estimation. They also found that the participants did not appear to use their declared rules of thumb during an experiment to estimate courseware development effort[4]. All of these effects have been widely reported in artificial intelligence literature in relation to expert knowledge elucidation (Diaper 1989).

The following are a few examples of the rules of thumb collected from participants in the previous studies:

- Nothing useful will happen in the first three to six months after you decide to go with [courseware]. It doesn't make any difference whether you go with a vendor or start producing your own in-house (Lee and Zemke 1987)
- The first course produced by a new [courseware] development group will be a collection of mistakes. Throw it away. (Lee and Zemke 1987)
- Where team members are not used to working together or are geographically apart, add 10-15% [to the total effort] (Casey et al. 1988)
- Analysis and design comprise 50% of total effort (Casey et al. 1988)
- Even a skilled Instructional Design author will revise plus or minus half the material after first or second draft (and then 20-25% in the third draft) (Casey et al. 1988)
- 50% of total design time is used for analysis/definition (Casey et al. 1988)

While individually the rules of thumb are of interest in terms of their contribution to general knowledge about courseware estimation, there has been no attempt to evaluate the accuracy of the estimates they produce. However, it may be worthwhile systematically investigating, collecting and encoding expert knowledge rules to develop an expert system-based courseware estimation tool. A classified list of rules of thumb

collected from the literature during this research can be found in Appendix 7.1.

Work breakdown

Work breakdown methods make use of a bottom-up approach to constructing estimates (Heemstra 1990; Heemstra 1992). The literature abounds with example projects and work breakdown templates for constructing estimates of multimedia courseware development effort (Bergman and Moore 1990; Imke 1991; Bunzel and Morris 1992). Table 2.3 shows an example work breakdown courseware development estimation template[5] based on Imke (1991). Work breakdown methods assume that estimating a number of smaller tasks is easier and, hopefully, more accurate than estimating the whole project. It is also hoped that by providing a template all the major tasks will be considered and nothing will be overlooked thus providing consistency in estimating. Unfortunately, there is very little evidence to support this premise specifically related to multimedia courseware development.

Table 2.3 - Work breakdown template (Imke 1991)

Stage	Description	[Effort (Dh)]	Charge Rate ($ Dh^{-1})	[Cost] Sub-total ($)
Plan	Project Leadership	70	60.00	4,200.00
Design	Project Leadership	80	60.00	4,800.00
	Course Development	200	60.00	12,000.00
	Instructional Design	40	60.00	2,400.00
	Video Production	24	80.00	1,920.00
	Programming	24	60.00	1,440.00
	Graphic Design	24	55.00	1,320.00
	Text Production	12	50.00	600.00
	Subject Matter Expertise	100	50.00	5,000.00
Specification	Project Leadership	48	60.00	2,880.00
	Course Development	12	60.00	720.00
	Instructional Design	24	60.00	1,440.00
	Video Production	24	80.00	1,920.00
	Programming	24	60.00	1,440.00
	Graphic Design	24	55.00	1,320.00
	Text Production	24	50.00	1,200.00
Script/Storyboards	Project Leadership	48	60.00	2,880.00
	Course Development	202	60.00	12,120.00
	Instructional Design	120	60.00	7,200.00
	Video Production	160	80.00	12,800.00
	Graphic Design	40	55.00	2,200.00
	Text Production	100	50.00	5,000.00
	Subject Matter Expertise	120	50.00	6,000.00
Video Production	Project Leadership	20	60.00	1,200.00
	Course Development	100	60.00	6,000.00
	Video Production	320	80.00	25,600.00
	Subject Matter Expertise	40	50.00	2,000.00
Program Production	Project Leadership	10	60.00	600.00
	Course Development	10	60.00	600.00
	Programming	100	60.00	6,000.00
	Graphic Design	100	55.00	5,500.00
Merge/Debug	Project Leadership	16	60.00	960.00
	Course Development	64	60.00	3,840.00
	Programming	80	60.00	4,800.00
	Graphic Design	64	55.00	3,520.00
	Text Production	12	50.00	600.00
Evaluation	Project Leadership	32	60.00	1,920.00
	Course Development	8	60.00	480.00
	Instructional Design	40	60.00	2,400.00
	Subject Matter Expertise	12	50.00	600.00
In-house sub-total		2572		159,420.00

The evidence that exists tends to indicate that the work breakdown methods do not produce accurate or consistent results. As part of a series of experiments Mikos et al. (1987b) explored the ability of courseware experts to use work breakdown approaches to estimate the development effort for a number of different delivery methods.

Participants were given detailed written descriptions of the courseware to be developed and were told to assume an "ideal" condition. A work breakdown template consisting of thirty-four tasks across the six phases of a well-defined Instructional Systems Development (ISD) life-cycle was provided to participants. The development effort for each task was summed to produce a Baseline Estimate (*BE*) for the whole project. Analysis of the sixteen computer-based courseware Baseline Estimates from the series of studies (Mikos et al. 1987a; Mikos et al. 1987b; Casey et al. 1988) produces a $\overline{MRE}$ of 0.41 and *PRED(0.25)* of 0.31 which clearly falls in the poor methods category (Campbell et al. 1988).

Work breakdown with project complexity adjustment

The second part of the Mikos et al. (1987b) experiment involved taking the Baseline Estimate (*BE*) described previously and amending it using a project complexity calculation tool. A Project Complexity Multiplier (*PCM*) was calculated from nine factors whose *combined* effect was thought to add effort to the project. Participants were allowed to make adjustments to the *PCM* until they were satisfied with the Adjusted Workload Estimate (*AWE*). The following equation shows how the *AWE* was calculated.

$$AWE = BE \times (1 + PCM)$$

Analysis of the results of 13 independent expert developers produces a $\overline{MRE}$ of 0.06 and a *PRED(0.25)* of 0.92 for instructor-lead training or computer-based courseware AWE. When only the 5 computer-based courseware estimates are considered the $\overline{MRE}$ drops to 0.12 and the *PRED(0.25)* to 0.80, which is still within the good classification for an effort estimation method (Campbell et al. 1988). The experiment was repeated at a conference with twenty four participants (Mikos et al. 1987b). In these sessions participants were asked to try to reach a consensus within the groups to which they had been assigned. A $\overline{MRE}$ of 0.53 and *PRED(0.25)* of 0.29 were produced which is in marked contrast to the results produced by the independent experts. Looking at the computer-based courseware results in isolation produced a $\overline{MRE}$ of 0.57 and *PRED(0.25)* of 0.45 which is still classified as poor.

The exercise was repeated a year later using workshop and video training scenarios (Casey et al. 1988). This time the twenty pre-conference experts produced a $\overline{MRE}$ of 0.31 and *PRED(0.25)* of 0.68 while the nineteen workshop participants produced a $\overline{MRE}$ of 0.79 and *PRED(0.25)* of 0.10. Again, both of these results fall clearly in the poor method category.

Table 2.4 summarises the results of the series of estimation experiments using work breakdown with project complexity.

Table 2.4 - Summary of work breakdown using PCM estimates

Delivery method	Number of Estimates	Participant Status	$\overline{MRE}$	*PRED(0.25)*	Method Accuracy
Courseware or Workshop	13	Independent	0.06	0.92	Good
Courseware only	5	Independent	0.12	0.80	Good
Courseware or Workshop	24	Group	0.53	0.29	Poor
Courseware only	11	Group	0.57	0.45	Poor
Workshop or Video	20	Independent	0.31	0.68	Poor
Workshop or Video	19	Group	0.79	0.10	Poor

These results appear to confirm Boehm's (1981) opinion that the use of any method based on expert judgement will produce results which are dependent on the participant's knowledge and skill. As shown in Table 2.5 the use of the *PCM* appears to have very little effect on the accuracy of the *BE* produced for the sixteen courseware estimates.

Table 2.5 - Comparison of BE and PCM for the sixteen courseware-only estimates

Estimation method	$\overline{MRE}$	*PRED(0.25)*	*PRED(0.30)*
Baseline Estimate (*BE*)	0.51	0.18	0.18
Project Complexity Multipliers (*PCM*)	0.57	0.45	0.55

While the *PCM* does slightly increase the $\overline{MRE}$ and has a significant effect on the *PRED(0.25)* and *PRED(0.30)* values, it still is not enough to upgrade the method from the poor classification. The net effect of using the PCM is to increase the proportion of good estimates by forcing the participants to consider factors which may increase development effort.

Scene counting

Philips Professional Media (1996) recommended the use of a scene-based or asset-based method for estimating the development *costs*[6] of CD-I multimedia titles including

courseware. The process by which the "end-user" requirements are specified into a story-board and its translation into programme material is called "title design". Philips Professional Media (1996) separated the provision of "content" from the "organisation" process and defined the production phase to have started when a clear specification of the title based on scene descriptor boxes as shown in Figure 2.3 has been produced.

Figure 2.3 - Example scene descriptor box (Philips Professional Media 1996)

Scene name	Description
Content	Thumbnail
Interactive links	

Based on these scenes, the *costs* associated with the provision of content and scripting can be built up. In addition, non-standard features such as simulations can be identified and the *costs* associated with them estimated. Table 2.6 shows an example CD-I costing worksheet.

Table 2.6 - Example CD-I costing worksheet (Philips Professional Media 1996)

Asset	Description	Quantity	Origination Price (£)	Conversion Price (£)	Total *[Cost]* (£)
Graphics	Photos	20	20	5	500
	Text Screens	50	20	5	1250
Video	3 minute Clips	10	3000	1200	42000
	10 minute Clips	2	10000	800	21600
Sound	Voiceovers	10	50	10	600
	Background Music	15	25	10	525
Scripts	Video Play/Control	10	5	0	50
	Timed to Voiceover	15	10	0	150
	Basic Selection	25	5	0	125
Menus	Video Play/Control	10	5	0	50
	Basic Selection	40	5	0	200
Total					67050

The method is based on a work breakdown method centred around analysis of the scene requirements which make up the courseware. While the method focuses on the *cost* of the project, it is fairly simple to amend it to produce estimates of effort. This would

have the advantage of being independent of the developer's charging scheme and the local currency, which is not true of the current *cost* method. The following equation could be used to provide a broad estimate for the total effort from the total *cost.*

$$Effort = \frac{Cost}{Developer\ effort\ charge}$$

In this case the *cost* is measured in *(£)* and *Developer effort charge (DEC)* is measured in *£-per-developer-hour (£ Dh^{-1})*. Alternatively, the original costing worksheet could be amended to estimate effort for each asset which could then be converted to *cost.*

There is no evidence of any systematic attempt to validate the method using actual courseware data and it appears to be based on internal experience at Philips (1996).

Industry averages

Jay, Bernstein and Gunderson (1987) indicated that "industry averages" are the most commonly used method to determine the effort[7] required to develop a unit of courseware. The "industry averages" method makes use of estimates of *effort to learner time ratios* and *learner time* to produce an estimate of effort required to produce the courseware. It is probably the oldest reported method for estimating courseware development effort. Dean and Whitlock (1983) stated that "Many statistics have been produced for the [effort] it takes, in [*developer hours*], to produce one [*learner hour*] of [*courseware*]". There is evidence of "industry average" being used as far back as 1967 for mainframe-based courseware effort estimation (Gerard 1967). Table 2.7 shows a small selection of *effort to learner time ratios* reported in recent literature.

Table 2.7 - "Industry average" *effort to leaner time ratios*

Courseware type	Author	Data collection method	*Mean effort to learner time ratios* ($\overline{Dh\ Lh^{-1}}$)	
			Lowest	Highest
Microcomputer-based	(Jones et al. 1993)	Author experience	200	1000
Microcomputer-based	(Miles and Griffith 1993)	Author experience	50	1000
Microcomputer-based	(Senbetta 1991)	Expert estimation	180	320
Microcomputer-based ICAL	(Orey et al. 1992)	Author experience	500	1000
Microcomputer-based ICAL	(Orey et al. 1994)	Experimental	16	81
Microcomputer-based Interactive Video	(Beautement 1991)	Author experience	> 400	
Microcomputer-based Multimedia	(Jay et al. 1987)	Expert estimation	1	4000
Microcomputer-based Multimedia	(Bourdeau et al. 1995)	Author experience	200	> 800
Microcomputer-based Multimedia	(Golas 1993)	Expert estimation	30	1390
Various	(Avner 1988)	Quasi-experimental	37	151

Table 2.7 seems to confirm Gery's (1987) view that "At conferences and in the press you often come across people quoting [*effort to learner time ratios*]. I have seen ratios of 25:1, 150:1 and 400:1. These ratios have been derived primarily through hearsay or some unsubstantiated 'industry average' and are nearly useless.". Analysis of Table 2.7 indicates the values cover a wide range of different types of courseware from simple drill and practice exercises (Senbetta 1992) through to high-fidelity multimedia simulations (Golas 1993). Similarly, there is a diverse range of delivery platforms spanning several computer generations. Some of the values have been derived from longitudinal studies lasting over 20 years (Avner 1979) whereas others are based on projects that lasted only a few months (Golas 1993). Other ratios appear to have been derived from the author's experience of the development process (Gery 1987; Beautement 1991; Orey et al. 1992; Jones et al. 1993; Bourdeau et al. 1995) whereas others are based on opinion distilled from groups of courseware development experts. These groups ranged in size from twenty (Senbetta 1991; Golas 1993) to one hundred and seventy nine (Jay et al. 1987) experts.

It is difficult to draw general conclusions from the range of values in Table 2.7 which limits the usefulness of "industry averages". Due to the diversity of multimedia courseware, "industry averages" on their own are too general to provide any more than the starting point for refinement by the developer (Gery 1987; Jay 1988). However,

"industry averages" do provide a starting point for a number of methods for estimating multimedia courseware effort which try to tailor the estimate to the courseware development environment.

2.2.3 Non-algorithmic methods

Five non-algorithmic methods for courseware effort or *cost* estimation were found in the literature. These methods make use of a diverse range of techniques to estimate multimedia courseware development effort or *cost*. The following five methods will be described briefly:

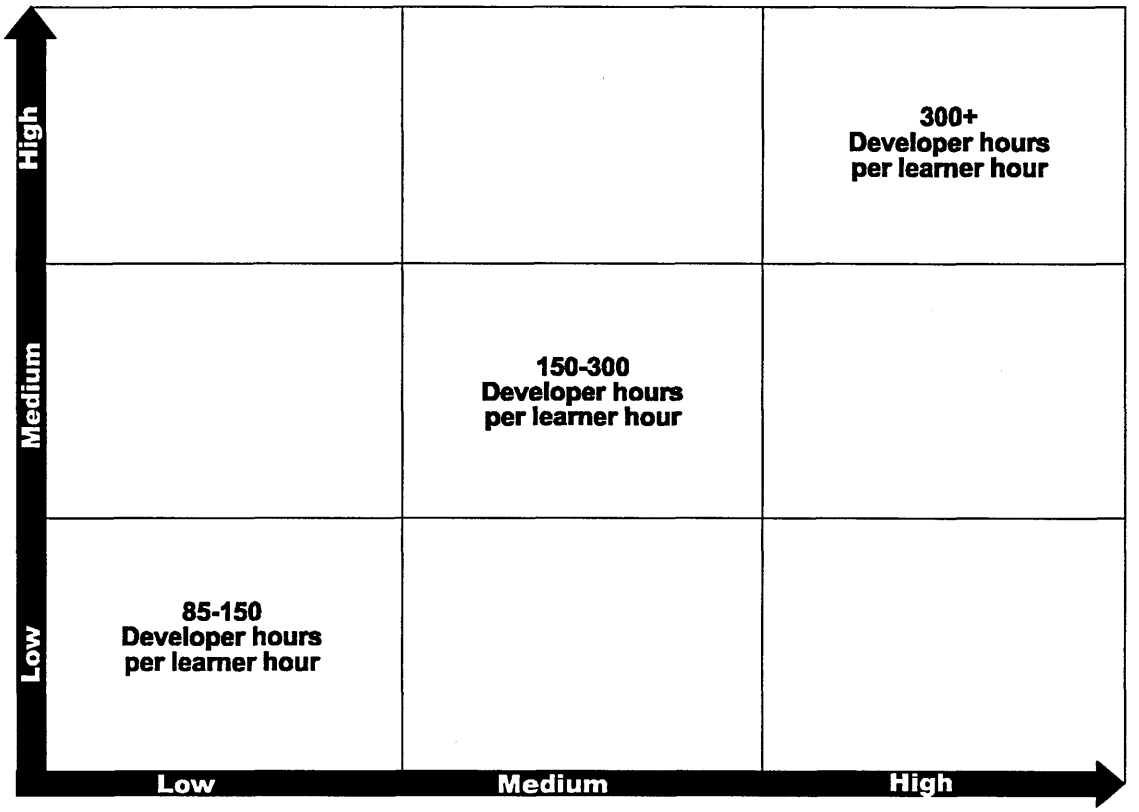
- Estimation grids
- Air force interactive courseware method
- 'Q' factor analysis
- Computer-based instruction length method
- Multimedia effort estimation method

While these could be converted into algorithmic or software based methods, they are presented in their original manual format.

Estimation grids

Gery (1987) outlined a method of estimating courseware *effort to learner time ratios*[8] using five estimation grids. Four of the grids show examples of sizing factors which affect different aspects of courseware *development time* and *cost*. Figure 2.1 shows the final grid which consists of three ranges of effort *to learner time ratios*. This final grid is used to identify the appropriate range of *effort to learner time ratios* from the estimator's analysis of the other four grids. An estimate of *learner time* is then multiplied by the appropriate *effort to learner time ratio* to produce a range of *effort* estimates for the courseware to be developed.

Figure 2.4 - Development *effort to learner time ratio* grid (Gery 1987)



The four other grids identify thirty-seven sizing factors which "... individually and collectively contribute to the total [effort] expended in courseware development" (Gery 1987). The thirty seven sizing factors are grouped under the following four headings:

- Courseware variables
- Technical variables
- Human variables
- Other variables

The method is a refinement of 'industry averages' which uses the thirty-seven sizing factors to tailor the estimate to the development conditions. Using the four broad groupings the estimator rates each of the thirty-seven sizing factors as low, medium or high in terms of its effect on *development time* and *cost*. An example of one of the four development variable grids and its associate sizing factors are shown in Figure 2.5 and Table 2.8 respectively.

Figure 2.5 - Relationship of courseware variables (Gery 1987)

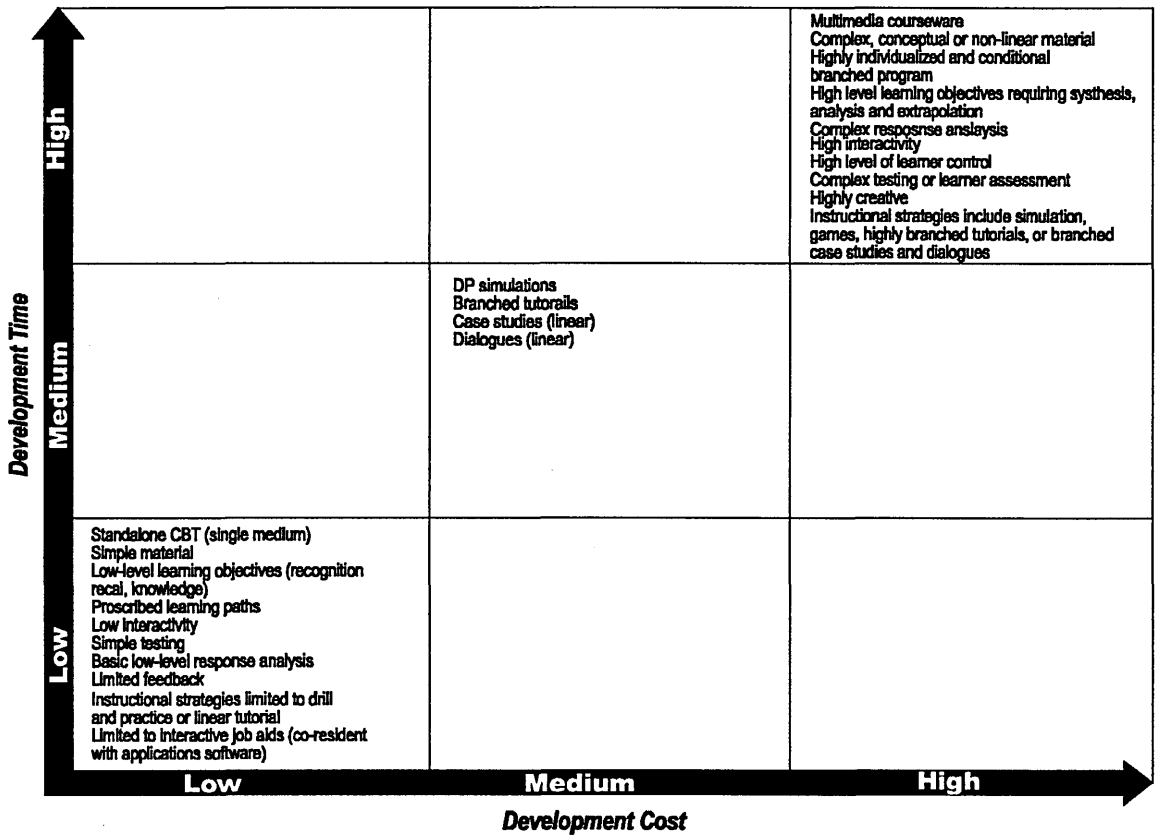


Table 2.8 - Courseware variables sizing factors (Gery 1987)

Description of courseware variables - sizing factors
Nature and complexity of learning material
Learning objectives (level)
Instructional design strategies
Nature and frequency of interactivity
Conditional branching
Response analysis: complexity, feedback and branching
Nature and depth of feedback
Nature and depth of testing
Creativity desired or required
Nature, complexity and volume of graphics and animation
Testing requirements
Courseware specification standards: quality, specificity and stability
Other media integration (type and complexity)
CMI (record-keeping and administration) requirements

Gery (1987) does not describe in any great detail how the method works. The estimator plots on the four courseware variable grids a low, medium or high rating of *development time* and *cost* for each of the thirty seven sizing factors. The estimator plots on the four development variable grids are then transposed onto the grid shown in

Figure 2.4 to produce a scattergram. Based on the clustering of the plots on the scattergram an appropriate range of *effort to learner time ratios* is selected by the estimator. This range of *effort to learner time ratios* can then be used along with an estimate of *learner time* to give a range for the estimated effort.

As can be seen from this brief description the method is informal and it is left to potential users to "... figure out how you will calculate the weight of each [sizing] factor..." (Gery 1987). Unfortunately, the method claims to produce estimates of *development time* and *cost* rather than effort. Gery (1987) stated that *development time* normally translates directly into *cost*, but numerous other variables, such as the choice of delivery medium affect the development *cost*. As with most other methods found in the literature the estimation grids method has not been evaluated. However, it does encourage the estimator to consider a number of sizing factors and tailor the estimate to the situation in which the courseware will be developed.

Air force interactive courseware method

Golas (1993) developed an estimation method for the US Air Force based on the opinion of twenty experts on the sizing factors which affect development effort[9]. The starting point is a best case estimate of the *effort to learner time ratio* for the level of course and type of behaviour to be delivered using the sizing factors listed in Table 2.9.

Table 2.9 - Best case *effort to learner time ratio estimate* (Golas 1993)

	[Effort to learner time ratio ($Dh\ \overline{Lh^{-1}}$)]		
Level of presentation	**Knowledge**	**Skill**	**Attitude**
I Basic	30	75	200
II Medium	75	125	250
III High	200	400	600

Basic level of presentation is defined to be introductory and linear in nature with little student interaction. The media used at this level are primarily simple graphics and text. A medium level of presentation normally involves more student control over the lesson presentation and a higher degree of interaction with the software. Simple records are kept of the student's performance. At this level the media used are primarily simple audio, video, graphics, animation and text. The high level of presentation is reserved for highly interactive presentations with extensive branching, real-time event simulation and interfacing to non-standard output devices. The media used at this level may

include full motion video, audio, complex animation, text and graphics.

The best case estimate applies when the thirteen sizing factors described in Table 2.10 are in place at the start of the project. For each sizing factor missing from a project, the best case estimate is increased by the appropriate *effort to learner time ratio* described in Table 2.10.

Table 2.10 - Sizing factors affecting best case estimates (Golas 1993)

Sizing factor	Increase [best case *effort to learner time ratio*] by: ($\overline{Dh\ Lh^{-1}}$)
1. No 'in-house' subject matter experts; must rely solely on the use of customer subject matter expertise	35
2. Subject matter is highly complex	100
3. Instructional content is unstable. Systems for which interactive courseware is being developed is emerging. Tasks for interactive courseware are constantly changing	100
4. Inadequate documentation. No training needs assessment was performed. No task analysis or learning analysis data. Technical manuals are non-existent or are not helpful	20
5. Total interactive courseware length less than 100 hours	20
6. Interactive courseware developer is not familiar with interactive courseware software/authoring systems	15
7. Interactive courseware developer is not familiar with target audience	10
8. Best commercial practices are not acceptable for video, graphics production and software development	50
9. Inexperienced project team: Interactive courseware designer inexperienced Interactive courseware manager inexperienced Interactive courseware programmer inexperienced	 80 100 60
10. Using a beta version of interactive courseware software	80
11. No prototype exists, no agreement "up front" on design strategy, no standardised development process followed	50
12. Customer is not using objective and consistent acceptance criteria. Customer unsure of what is wanted and does not communicate with developer	50
13. Required resources are not in place at start of project	20

Based on the product of the *effort to learner time ratio* and the estimate of *learner time* the effort required to undertake the project can be estimated. Golas (1993) has extended the method provided by "industry averages" and provided developers with guidance on how to tailor their original estimates to suit the circumstances of the project. There is no evidence that this method has been externally validated using actual courseware data. This can be difficult to arrange as Golas (1994) stated "... many companies in the Interactive Multimedia market keep data on [effort], but most of them are reluctant to

give it out. The twenty reviewers who commented on my paper looked at their own raw data when they made their comments."

'Q' factor analysis

Bergman and Moore (1990) drew an analogy between estimating multimedia development *cost*[10] and the method used in house building of quoting cost per unit area ($/sq. ft) through the use of Quality or *'Q'* factors. The following equation show how the *Q* factor is calculated.

$$Q = \frac{Cost\ /\ 100{,}000}{Learner\ time}$$

To keep *Q* small the *Cost* is divided by 100,000. From personal experience the authors indicated that *Q* ranges from 0.5 to 1.5 for the preparation of the first training module although it is possible for the factor to rise up to 5 or higher in special cases. In common with the house building analogy, the higher *Q*, the higher the quality of the finished product. Table 2.11 gives examples of *Q* for different types of multimedia projects.

Table 2.11 - 'Q' factors for example multimedia projects (Bergman and Moore 1990)

Description	'Q' factor ($ Lh^{-1})	Example 1 ($\overline{Dh\ Lh^{-1}}$)	Example 2 ($\overline{Dh\ Lh^{-1}}$)
Long timescale project - custom training	0.1	50	100
Short timescale project - simple custom training	0.5	250	500
Short timescale project - complex custom training	1.5	750	1500
Simple public access and marketing presentation	2.5	1250	2500
Complex public access and marketing presentation	5	2500	5000
Video-rich application using professional facilities	> 1	500	1000
Video-rich application using professional facilities and actors	> 3	1500	3000
Video-rich application using professional facilities and external locations or celebrity actors	5	2500	5000

Once the quality of the project is established by selecting *Q* the *cost* of a multimedia project can be estimated using the following equation.

$$Cost = Q \times 100{,}000 \times Learner\ time$$

Bergman and Moore (1990) only considered the *cost* of development; however, it is fairly simple task to convert the *cost* into effort using following equation.

$$Effort = \frac{Q \times 100{,}000 \times Learner\ time}{Developer\ effort\ charge}$$

The *Developer effort charge (DEC)* is effectively the rate charged to clients. It encompasses payroll and non-payroll cost as well as the developer's overheads and profit and is measured in *$ per developer hour ($ Dh^{-1})* The example effort *to learner time ratios* shown in Table 2.11 are based on nominal *Developer effort charge* of 200 *$ Dh^{-1}* in Example 1 and 100 *$ Dh^{-1}* in Example 2 (Schooley 1988; Imke 1991; Bunzel and Morris 1992). These examples show that the estimated *effort to learner time ratios* are of similar magnitude to those described by other authors (Gery 1987; Jay et al. 1987; Golas 1993). Bergman and Moore (1990) are experienced developers of multimedia projects; their method appears to be based on their own data set and there is no evidence that the method has been validated using external data.

Computer-based instruction length method

Tan and Nguyen (1993) proposed the computer-based instruction length method (CBIL)[11] from a total development and delivery life-cycle cost-benefit analysis perspective. CBIL is intended to estimate not only the development *cost* but also the hardware capital and other recurring expenditure over the normal life of the courseware. It therefore goes much further than required for this research but the development estimation element has been extracted from the underlying method. CBIL differentiates three distinct modes of courseware development based on the delivery platform requirements and courseware facilities. Table 2.12 describes the three categories of courseware development used by CBIL.

Table 2.12 - CBIL courseware categories (Tan and Nguyen 1993)

CBIL mode	Hardware requirements	Courseware facilities
1	Standard hardware platform with keyboard and mouse and VGA screen	Text, graphics and simple animation but no stills or motion video
2	As for CBIL-1 plus sound and video facilities such as laserdisc and video or TV playback through a 'video toaster'	Text, graphics, simple animation, stills and motion video
3	As for CBIL-2 plus additional input through touch screen or microphone and output to projection TV or printers, CD-ROM and other digital video facilities	Text, graphics, complex animation, stills and motion video

CBIL claimed to use the following equation to estimate the total development *cost* (D_i) where i is mode 1, 2 or 3.

$$D_i = m_i M_i + j_i J_i + a_i A_i + G_i g_i + s_i S_i + v_i V_i + p_i P_i + o_i O_i$$

Where:

M_i = Material preparation costs

J_i = Instructional design costs

G_i = Graphical design and production costs

A_i = Authoring and testing cost

S_i = Scripting costs of the video and audio production

V_i = Video and audio production costs

P_i = Video and audio post-production costs

O_i = Optical disc mastering and duplication costs

The lower case variables are parameters which depend on the type of hardware and software and associated personnel costs. Table 2.13 shows an example which has been constructed using the developer and media estimation template.

Table 2.13 - CBIL developer and media *cost* estimation template (Tan and Nguyen 1993)

Basic costing assumptions			**Value**
Instructor-led classroom course unit length			1 hour
Number of course units			20
Courseware developer hourly rate			A$30
Costing assumptions for each mode	**CBIL-1**	**CBIL-2**	**CBIL-3**
Average *[effort to learner time ratio]*	18:1	60:1	180:1
[Courseware] compression	70%-50%	50%-30%	30%-0%
Estimated effort per course unit			
Material preperation	6	6	24
Instructional design	6	6	24
Graphics design	0	12	24
Graphics production	0	18	24
Authoring and testing	6	18	40
Scripting	0	0	24
Project management	0	0	20
Estimated *effort* per course unit	18	60	180
Estimated *effort* for total course	360	1200	3600
Total developer cost for course	A$10800	A$36000	A$108000
Estimated media cost per course unit			
Video production (6 minutes)	0	0	A$3000
Audio production (30 minutes)	0	0	A$1500
Post production	0	0	A$1000
Disk mastering	0	0	A$3000
Media cost per course unit	0	0	A$8500
Total media cost for course	0	0	A$170000
Total courseware development cost	**A$10800**	**A$36000**	**A$278000**

Despite the claim to use the *cost* estimation formula, the method appears to be a variation on a work breakdown template with predefined values for certain activities under the three different courseware modes. However, Tan and Nguyen (1993) used the correct units to measure effort ("person hours") and *cost* ("Australian Dollars"). Unfortunately, there is no evidence of internal or external validation of the method proposed.

Multimedia effort estimation method

The multimedia effort estimation method was developed by Marshall et al. (1994c) to explore the potential of using software metrics techniques in the development of multimedia courseware effort estimation models[12]. The method was based on regression analysis of courseware development data from 14 projects. The resulting method was able to explain 85% of the variation in development effort using the following equation.

$$Effort = \alpha\,(Learner\ time)^{\beta} \times SD(X)$$

Where *learner time* measures courseware size, α is the dimensionality coefficient and β is an economy of scale coefficient. In MEEM both α and β were given the value of one. The dimensionality coefficient α is measured in units of $Dh\,Lh^{-1}$ to convert *learner time* measured in *Lh* to effort measured in *Dh*. The function of *SD(X)* is to tailor the estimate to the circumstances which exist in the courseware development environment. The value of *SD(X)* was calculated using the following four grouped super drivers.

- Course difficulty (CD)
- Interactivity (IN)
- Development environment (DE)
- Subject expertise (SE)

These super drivers were calculated by summing the values assigned to individual cost drivers which formed each of the four groupings. Each of the individual cost drivers must be rated using the rating scale. If a particular cost driver does not apply to the courseware being estimated then it is given the lowest rating value of one.

The course difficulty (CD) super driver brings together three individual cost drivers which are intended to describe the effect of the complexity or difficulty of the subject matter on the effort required to develop the proposed courseware development. Table 2.14 shows the three course difficulty cost drivers used in MEEM.

Table 2.14 - Course difficulty cost drivers (Marshall et al. 1994c)

CD rating scale	1	2	3	4	5
Number of course objectives	Less than 20	21 to 40	41 to 60	61 to 80	Greater than 80
Level of course objectives	Concrete concepts	Abstract concepts	Lower order principles	Higher order principles	Problem solving
Existing course material	Rewrite of existing multimedia material	Rewrite of existing [courseware] material	Rewrite of written material	Rewrite of Tutor delivered material	New course

Interactivity (IN) brings together fourteen individual cost drivers related to the use of media and interactivity of the courseware to be developed. The fourteen individual interactivity cost drivers used in MEEM are shown in Table 2.15.

Table 2.15 - Interactivity cost drivers (Marshall et al. 1994c)

IN rating scale	1	2	3	4	5
Complexity of interface	Simple text-based	Complex text-based	Simple graphics-based	Complex graphics-based	Windowing graphics-based
Level of interactivity	Linear	Simple branching	Complex branching	Simple adaptive	Complex adaptive
Type of question feedback	None	Right/wrong	Right/wrong with right feedback	Right/wrong with relevant feedback	Right/wrong with remediation and feedback on each wrong answer
Majority question style	True/false	Multiple choice	Single words	Limited free text	other
Graphics requirements	None	Existing artwork	Simple original artwork	Complex original artwork	Extremely complex artwork
Graphics density	Less then 1 per 20 frames	1 per 20 frames	1 per 10 frames	1 per frame	More than 1 per frame
Animation requirements	None	Existing animation	Simple animation	Complex animation	Mathematically accurate animation
Animation density	Less then 1 per 20 frames	1 per 20 frames	1 per 10 frames	1 per frame	More than 1 per frame
Audio requirements	None	Existing audio	Simple original audio	Complex original audio	Extremely complex audio
Audio density	Less then 1 per 20 frames	1 per 20 frames	1 per 10 frames	1 per frame	More than 1 per frame
Video requirements	None	Existing video	Simple original linear video clips	Complex original linear video clips	Complex original interactive video clips
Video density	Less then 1 per 20 frames	1 per 20 frames	1 per 10 frames	1 per frame	More than 1 per frame
Simulation requirements	None	Existing simulation	Simple original simulation	Complex original simulation	Realistic simulation
Simulation density	Less then 1 per 20 frames	1 per 20 frames	1 per 10 frames	1 per frame	More than 1 per frame

The development environment (DE) super driver brings together five individual cost drivers which describe the effect of development tools and team expertise on the effort required to develop the proposed courseware. Table 2.16 shows the five development environment cost drivers used in MEEM.

Table 2.16 - Development environment cost drivers (Marshall et al. 1994c)

DE rating scale	1	2	3	4	5
Production environment	Authoring environment	Authoring system	Authoring language	High level language	Low level language
Instructional design, development and delivery methodology	Formal third generation	Formal second generation	Formal first generation	Informal	None
Size of proposed development team	1	2-4	5-9	10-15	More than 15
Development team's subject matter experience	Expert knowledge of the subject	Good knowledge of the subject	Some knowledge of the subject	Knowledge of related subject	No knowledge of subject
Development team's multimedia experience	Extensive multimedia development experience	Some multimedia development experience	Extensive [courseware] experience	Some [courseware] experience	None

Subject expertise brings together two individual cost drivers related to the availability of the subject matter experts and their experience of creating courseware. The two individual subject expertise cost drivers used in MEEM are shown in Table 2.17.

Table 2.17 - Subject expertise cost drivers (Marshall et al. 1994c)

SE rating scale	1	2	3	4	5
Subject matter expert's multimedia experience	Extensive multimedia development experience	Some multimedia development experience	Extensive [courseware] experience	Some [courseware] experience	None
Availability of subject matter expert	Unrestricted contact	Daily contact	Weekly contact	Monthly contact	Restricted contact

Using just the data from the ten developments with *learner time* equal to one hour and one outlier removed the following regression equation on CD and DE was found to explain 85% of the variance.

$$Effort = \alpha\,(Learner\ time)^{\beta} \times (120CD + 30.7DE - 1413)$$

The inclusion of IN and SE in the regression did not improve the result. This may be because CD and IN are highly correlated at 0.65 and there is a very limited range of SE values produced with this data set. This resulting effort estimation equation produced an acceptable rating for a courseware effort estimation method with $\overline{MRE} = 0.27$ and $PRED(0.30) = 0.70$ (Campbell et al. 1988).

However, MEEM has a number of limitations. The first and most serious is that it can produce zero or negative effort under certain conditions. For example, if CD is equal to

its minimum value of three then DE cannot take a value less than twelve. Similarly, if CD is equal to its maximum value of fifteen then DE cannot take a value less than eight. The second limitation is that the relatively small size of the data set makes it difficult to generalise these results. However, Marshall et al. (1994c) did not suggest that this was a general model. They felt that it indicated the potential for further research and that further project data would be required to validate and evaluate the model.

2.2.4 Algorithmic methods

Six algorithmic or software-based methods for courseware estimation were found in the literature. The following six methods will be briefly described:

- CBT Analyst
- Cost estimating algorithm for courseware
- Quick and dirty method
- Regression method
- Training cost calculation spreadsheet
- System dynamics model

These make use of sizing factors which have been identified as potentially increasing or decreasing the development *effort* or *cost* of multimedia courseware.

CBT Analyst

Kearsley's (1985) CBT Analyst is a courseware tool which works by asking the developer questions about the courseware to be developed. The software produces advice for developers on selecting, developing and costing courseware. In addition, it attempts to predict the benefits and likely success of using the courseware. The tool also allows individual developers to add or amend the rules on which the software is based to tailor it to match the developer's experience. One of the five elements of the software produces *effort to learner time ratios*[13] using the base constraints identified in the twenty-two questions described in Table 2.18.

Table 2.18 - CBT Analyst's base constraints questions (Kearsley 1985)

Question	Lowest Value	Highest Value
1. What type of [courseware] do you plan to develop? (tutorial, simulation, testing or embedded)	0	+5
2. How complex is the learning task for which the [courseware] is to be developed?	0	+2
3. Will colour graphics be used?	0	+5
4. Will interactive video or audio be used?	0	+5
5. How will the courseware be developed?	0	+3
6. Does a library of [courseware] routines and graphics exist or does all programming have to be done from scratch?	-5	0
7. How much [courseware] experience does the designer or design team have?	+1	+5
8. How much experience does the developer/programmer have with the authoring language or system being used?	-5	0
9. Is this a new or existing course?	0	+5
10. Is the subject matter for the course available or is it in the process of being developed?	0	+5
11. Is the [courseware] being developed for internal use or will it be sold commercially?	0	+5
12. What kind of branching will the course involve?	0	+5
13. Will the answer analysis be simple or complex?	0	+5
14. What kind of response will the course involve?	0	+5
15. How much learner control will the program have?	0	+5
16. What percentage of the course do you anticipate having to revise each year?	0	+3
17. Does a well defined storyboard exist for the [courseware] to be developed?	-5	0
18. If the [courseware] is to be developed by a team, does this team have previous experience developing [courseware] courses together?	0	+5
19. Do written standards, guidelines, or procedures exist for [courseware] development and are they followed?	0	+5
20. Is the development effort being managed by an individual with past experience of managing [courseware] projects?	0	+5
21. Is there a single individual responsible for approving the course and revisions to be made?	0	+5
22. How would you describe the motivational level of the designer/developer?	-10	0

In Table 2.18 a positive value indicates the base constraint increases development effort whereas a negative values decreases the development effort. Table 2.19 describes composite rules which are then used to modify some of the base constraint results.

Table 2.19 - CBT Analyst's composite rules (Kearsley 1985)

Composite rule	"Unknown" rating in questions	New score
Rule 32 - Inadequate [courseware] specification	1, 13 and 15	+10
Rule 33 - Human factors unknown	20,21 and 22	+10
Rule 34 - Experience unknown	7 and 8	+5

Finally, CBT Analyst totals the base constraints and uses this to select an estimated range of *effort to learner time ratio* using the threshold values in Table 2.20.

Table 2.20 - CBT Analyst's threshold values and effort to learner time ratios (Kearsley 1985)

Threshold values	Range of [*effort to learner time ratio* ($\overline{Dh\ Lh^{-1}}$)]
-9999 to 0	Under 100
1 to 20	100 - 200
21 to 50	200 - 400
51 to 9999	500+

There has been some criticism of the selection and the relevance of individual base constraints together with the range and step size of the final estimate. (Jay et al. 1987; Jay 1988) However, it does provide a consistent and repeatable estimating method which is simple to use. Jay, Bernstein and Gunderson (1987) comparing the results from CBT Analyst against data from nine projects claimed that it produced six estimates ranging from 86% too low to 70% too high and that three estimates were within 10% of actual effort *to learner time ratio*. Unfortunately, as Table 2.21 shows, re-analysis of the results failed to reproduce these figures.

Table 2.21 - CBT Analyst relative error for actual project data (Jay et al. 1987)

Project Number	Actual *effort to learner time ratio* ($\overline{Dh\ Lh^{-1}}$)	Estimated *effort to learner time ratios* ($\overline{Dh\ Lh^{-1}}$)			Absolute Relative Error		
		Low	Mean	High	Low	Mean	High
1	117.5	200	300	400	0.70	1.55	2.40
2	220.5	100	150	200	0.55	0.32	0.09
3	972.0	100	150	200	0.90	0.85	0.79
4	266.2	200	300	400	0.25	0.13	0.50
5	270.8	200	300	400	0.26	0.11	0.48
6	211.1	200	300	400	0.05	0.42	0.89
7	294.2	500	750	1000	0.70	1.55	2.40
8	384.0	100	150	200	0.74	0.61	0.48
9	312.5	100	150	200	0.68	0.52	0.36

Project number seven originally reported an estimate of 500+ $\overline{Dh\ Lh^{-1}}$ with no upper limit. The value of 500 $\overline{Dh\ Lh^{-1}}$ is selected as the lower limit while a value of 1000 $\overline{Dh\ Lh^{-1}}$ is used as the upper limit on the basis that all the other ranges of estimate produced by CBT Analyst use this doubling technique. As Table 2.21 shows, only two of the relative errors for the low, high and mean estimates of effort *to learner time ratios* are within 10% (0.10). Table 2.22 show the $\overline{MRE}$ and *PRED(l)* results for the re-analysis.

Table 2.22 - Predictive accuracy of CBT Analyst (Jay et al. 1987)

Measure	Low Estimate	Mean Estimate	High Estimate
$\overline{MRE}$	0.54	0.93	0.67
PRED(0.20)	0.11	0.22	0.33
PRED(0.25)	0.11	0.11	0.11
PRED(0.30)	0.22	0.22	0.22

The method clearly falls within the poor classification for a courseware effort estimation method (Campbell et al. 1988). However, it is hardly surprising the method produces such poor results because it was not calibrated in any way to the data which was provided by nine independent project managers. It is therefore perhaps too early to disregard CBT Analyst entirely.

Cost estimating algorithm for courseware

Schooley's (1988) Cost estimating algorithm for courseware (CEAC) method estimates courseware development *effort* and *cost*[14]. The software requires the estimator to enter fourteen Course Independent and seventeen Course Dependent Parameters which it uses to make an estimate. Table 2.23 lists the Course Independent and Course Dependent Parameters used in CEAC.

Table 2.23 - CEAC course independent and course dependent parameters (Schooley 1988)

Parameters	Main component	Parameter name	Example
Independent	Cost primitives and derived costs	Overhead rate	110%
		General and administrative rate	15%
		Fee	10%
		Category I labour rate	$25 Dh^{-1}
		Category II labour rate	$20 Dh^{-1}
		Category III labour rate	$15 Dh^{-1}
		Category IV labour rate	$10 Dh^{-1}
	Pedagogical base	Lecture-to-[courseware]-advantage ratio	0.75
	Productivity tools	Tutorial	C
		Drill and practice	C
		Simulation	B
		Certification test	C
	Teaming multipliers	Team size	3
Dependent	Pedagogical base	Lecture material exists	Y
	Delivery time estimate	Equivalent for lecture-based	10 hours
	Team experience	Experience factor	1.4
	Labour category	Number	I
		Percentage contribution	0.25
	Library availability	Utility library	10%
		Application library	20%
		Graphics library	5%
		Character set library	5%
	Functional mode break down	Tutorial	40%
		Drill and practice	20%
		Simulation	30%
		Certification test	10%
	Instructional sophistication	Tutorial	0.6
		Drill and practice	1
		Simulation	1
		Certification tests	2

The software uses organisational and courseware specific sizing factors and data derived from analysis of a limited set of sample projects to produce estimates. CEAC's internal database makes use of courseware development data provided by the author and from published sources (Avner 1979) to construct cornerstone values. These cornerstone values assume the courseware to be developed will have a non-existing pedagogy, an inexperienced team for each of the functional modes and will use Class 'A' productivity tools. All other data points in the database are extrapolated linearly from these cornerstone values.

Based on the estimator's inputs and its internal database, CEAC uses the following equation to calculate the effort required to develop the courseware by summing the contribution of tutorial, drill and practice, simulation and certification test elements to

the total project.

$$Effort = \sum (CF \times CA \times LT) \times (TM \times EF \times SF \times DV \times [1 - LSF])$$

Where:

CF	= Courseware fraction	EF	= Experience factor
CA	= Courseware advantage	SF	= Sophistication factor
DV	= Database values	LSF	= Library-saving fraction
LT	= Lecture equivalent time	TM	= Teaming multiplier

The *developer effort charge*$_i$ *(DEC*$_i$*)* is calculated for each of the four categories of labour using the following equation.

$$Developer\ effort\ charge_i = CL_i \times (1 + OR_i) \times (1 + GA_i) \times (1 + FE_i)$$

Where:

OR_i	= Overhead rate	FE_i	= Fee rate
GA_i	= General and administrative rate	CL_i	= Category labour rate

The *cost* is calculated by summing the individual contribution of the four categories of labour using the costing data previously entered and the following equation.

$$Cost = \sum_{i=1}^{4} \left(Effort \times DEC_i \times Number_i \times Percentage\ of\ development_i \right)$$

Where $Number_i$ is the total number of developers allocated to the project for labour category i and $Percentage\ of\ development_i$ describes the percentage of the development allocated to this labour category

Schooley (1988) discussing the results of an informal validation of CEAC using data from twelve projects reported a result equivalent to a *PRED(0.20)* of 0.50. Unfortunately, the $\overline{MRE}$ was not directly reported and analysis of the graphical results produced the range of estimates for $\overline{MRE}$ and *PRED(l)* shown in Table 2.24.

Table 2.24 - Estimates of CEAC's accuracy

Measure	Low Estimate	Middle Estimate	High Estimate
$\overline{MRE}$	0.34	0.31	0.27
PRED(0.20)	0.25	0.25	0.58
PRED(0.25)	0.58	0.58	0.58
PRED(0.30)	0.58	0.58	0.67

Whatever the value of CEAC's actual $\overline{MRE}$, the estimates suggest that it is likely to fall into the poor method range with a $\overline{MRE} > 0.30$ and *PRED(0.30)* < 0.70. Despite this CEAC has a number of strengths; it uses a range of courseware and organisational factors that contribute to the final estimate of courseware *development time*, *effort* and *cost*. Unfortunately, a number of the sizing factors used by CEAC appear to have been constructed from linear projections from a limited number of data points which may account for its poor performance.

Quick and dirty method

Miles and Griffith (1990b; 1993) developed a Lotus 123-based template to estimate the *effort to learner time ratio*[15] required to produce courseware across four phases of a development life-cycle. Both Miles and Griffith are experienced courseware developers working for commercial companies which depend on the accuracy of their estimates to stay in business. The model is intended to produce "quick and dirty" estimates and as the estimators become more experienced, the figures in the spreadsheet should be amended to improve the accuracy.

The basic method used six sizing factors to identify the constraints for a given project. A score is generated by the estimator's answers to questions about each of the six sizing factors. These values are then summed for each phase and a final calculation is used to estimate the *effort to learner time ratio* for individual phases and the project as a whole. The basic template assumes a basic *effort to learner time ratio* of 300 *developer-hours-per-learner-hour* for level 1 courseware (US Navy 1987). Miles and Griffith (1993) indicated that this basic *effort to learner time ratio* can be increased if required. The method also assumes that 40% of the effort is devoted to analysis, 20% to design, 25% to development and 15% to implementation. Table 2.25 describes the basic method which covers the whole courseware development life-cycle.

Table 2.25 - "Quick and dirty" development template (Miles and Griffith 1993)

Variables	Score	Phase 1 Analysis	Phase 2 Design	Phase 3 Develop	Phase 4 Produce
Task complexity					
Simple	-1				
Average	0				
Complex	2				
Highly complex	4				
[Courseware] level					
Level 1	0				
Level 2	1				
Level 3	3				
Development System					
Sophisticated authoring system	-1				
Limited authoring system	0				
Authoring language	2				
Programmed language	4				
People					
Inexperienced	1				
New team (<1 year)	1				
Old team (>2 years	-1				
Separate locations	1				
2 or 3 of above	1				
Existing materials					
Yes	-1				
Some	0				
No	1				
Existing standards					
Yes	-1				
No	1				
Number of Marks					
Muliplied by		12	6	7.5	4.5
Equals					
Plus		120	60	75	45
Total developer hours per phase					
Estimate for all phases					

While Miles and Griffith (1990b; 1993) indicated that the method "... had been used on several projects, with a high degree of agreement between the estimated and actual [effort]", no published examples have been released because of the commercial confidentiality of the information. In common with most of the other methods described in this Section there is no published evidence of independent validation.

Regression method

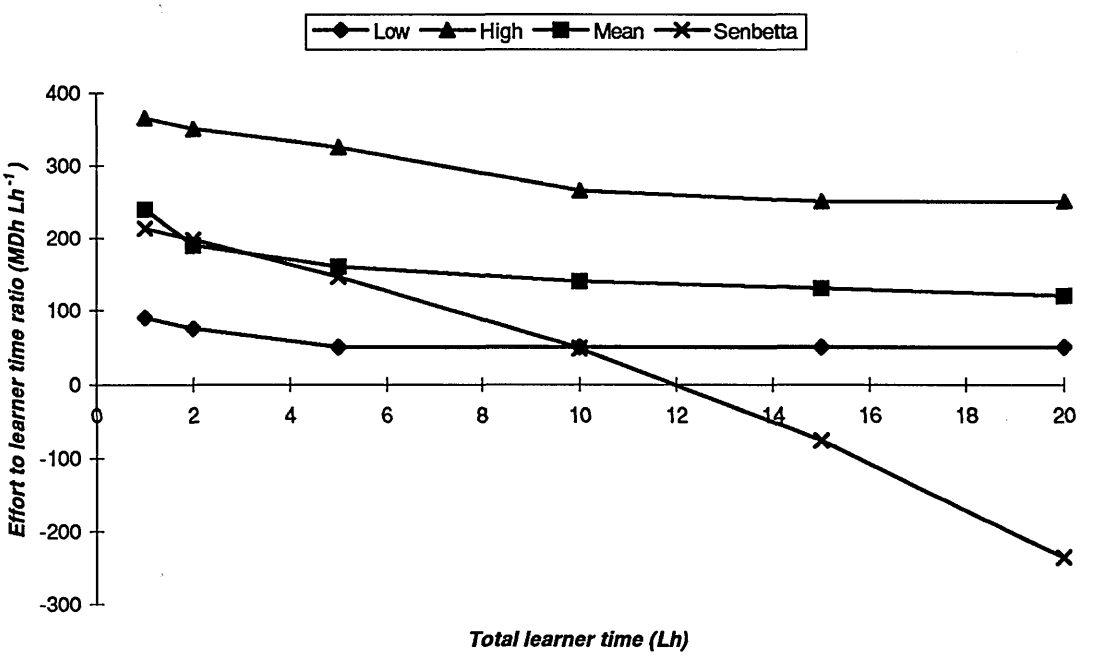
A polynomial regression equation for predicting courseware *effort to learner time ratios*[16] was proposed by Senbetta (1991; 1992). The method was based on the analysis of a Delphic study of the opinion of twenty five expert courseware developers about the

effort to learner time ratio required to produce the first, second, fifth, tenth and twentieth *learner hour* of a well specified courseware development project. The following equation attempts to predict the *effort to learner time ratio* required to produce (t) *learner hours* of courseware.

$$\textit{Effort to learner time ratio} = 0.000176t^4 - 0.210t^3 - 16.308t + 230.102$$

Senbetta (1992) claimed that the equation would only be as good as the assumptions made and the data available. Unfortunately, he did not publish the percentage variance explained by the method (R^2) and, as Figure 2.6 shows, a serious error appears to have been made in the regression analysis and the equation is wrong.

Figure 2.6 - Predicted *effort to learner time ratio* graph (Senbetta 1991)



The choice by Senbetta (1992) of polynomial regression to explain the data is questionable. Polynomial expressions have localised turning points with the possibility of negative *effort to learner time ratios* for positive values of *learner time*. Regression analysis of Senbetta's (1992) data for the six mean data points produces the following equation for *effort to learner time ratio* ($R^2 = 0.997$).

$$\textit{Effort to learner time ratio} = 243.64t^{-0.3913}$$

The resulting equation is in the form of a classical learning curve which predicts the effort per unit of production required to produce products as a process matures. It is normally deemed appropriate in processes with a high degree of manual skill. The learning curve predicts the reduction in effort required to produce an individual product in a learning situation (Woolf et al. 1985). The available evidence indicates this is the situation which occurs in the production of courseware. The first hour of *learner time* in a series of related courseware modules will consume more effort then the last (Avner et al. 1984). In the early stage of a long courseware development style conventions, templates and tools are developed which will be used in later modules.

Training cost calculation spreadsheet

Milette and Trevor-Deutsch (1995) prepared a spreadsheet-based worksheet to produce a cost-benefit analysis of classroom-based versus courseware-based training. The worksheet, while primarily concerned with comparative costs, does make estimates of the *learner time*, *effort to learner time ratio*, *development time* and *cost*[17] associated with both classroom-based and courseware-based training. Table 2.26 shows the costing worksheet used with the spreadsheet.

Table 2.26 - Example costing worksheet (Milette and Trevor-Deutsch 1995)

Fill in as many components in the guide as you can. Do not fill in shaded components.		Run number			
A	**Data Entry Components**	1	2	3	4
1	Estimates and Assumptions (Real=0, Loaded=1)				
2	Estimated Course Length for Classroom Version (hrs.)				
3	Course Length for [Courseware] vs Classroom Instruction				
4	Instructor/Developer Labour Cost per Hour				
5	Estimated Total number of Students				
6	Estimated Number of Students per Class				
7	Course Development Ratio for Classroom Instruction				
8	Estimated Materials Cost Per Student				
B	**[Courseware] Development Weighting Factors**				
9	Lesson Style (Program Complexity): page turner=1, tutorial=5 simulation=10				
10	Graphics: text only=1, functional=5, animation/simulation=10				
11	Course Developer Experience: 3+ courses =1, 2-3 courses=5, 1st course=10				
12	Programmer Experience: many templates=1, limited=5, 1st course=10				
13	Critical Path - Graphics: graphics done before=1, during=5, after=10				
14	Software Sophistication: simple=1, basic features=5, complex=10				
15	Peripherals: none=1, regular devices=5, specialised (eg video)=10				
16	Development: working from existing materials=1, creating new=10				
17	Quality Needed: low=1, commercial=10				
18	Size of Product: small=1,medium=2, large=10				
19	Degree of Remediation: low=1, medium=5, high=10				

The method makes a number of assumptions including using a maximum effort *to learner time ratio* of 400 $Dh\ \overline{Lh^{-1}}$ although there is a warning that this "... would NOT be accurate for multimedia production where the ratio can go much higher" (Milette and Trevor-Deutsch 1995). Using an average of the eleven courseware development weighting factor (*CDWF*) values entered in Table 2.26, the spreadsheet calculates an *effort to learner time ratio* for the project using the following equation.

$$\textit{Effort to learner time ratio} = \left(\frac{\sum_{i=9}^{19} CDWF}{11} \right) \times \frac{400}{10}$$

This gives a range of *effort to learner time ratios* between 40 $Dh\ \overline{Lh^{-1}}$ and 400 $Dh\ \overline{Lh^{-1}}$ depending on the *CDWF* used. The value of *learner time* is calculated by multiplying the estimated course length for the classroom-based version by the course length for courseware vs classroom instruction value. An underlying assumption is that the courseware instruction will be more efficient than the classroom equivalent. In the worksheet a 50% *courseware* compression value is used. Milette and Trevor-Deutsch

(1995) claims that a value as low as 25% could be used for the course length for courseware vs classroom instruction value. While the 50% value has support in the literature, the reduction of *learner time* to just 25% of the classroom-based course length appears extreme (Orlansky and String 1979; Kulik and Kulik 1986; Lisewski and Settle 1995).

In common with most authors, Milette and Trevor-Deutsch (1995) incorrectly use the *effort to learner time ratio* to calculate *development time* rather than effort. The corrected equation is shown below.

$$Effort = Effort\ to\ learner\ time\ ratio \times Learner\ time$$

The reason for the confusion lies in the unstated assumption that only one developer will be used on the project. In this special case, effort and *development time* do yield the same numeric value but they should still be treated as separate metrics with their own distinct units. Given this problem, the *cost* of the courseware development is then estimated using the following equation.

$$Cost = Effort\ to\ learner\ time\ ratio \times Developer\ effort\ charge \times Learner\ time$$

Similar calculations are carried out for the classroom-based instruction along with estimation of instructor involvement in delivery of both methods to enable a cost-benefit analysis to be carried out using tabular and graphical reports. The method appears to be based on the author's experience of developing training materials and there is no internal or external validation using actual project data. Despite these problems, the method does attempt to tailor the estimates produced using a small number of sizing factors related to courseware development. It is also relatively simple to use and can be customised based on the estimator's experience.

System dynamics model

The Grimstad Group (Davidsen et al. 1995) have started to explore the use of system dynamics (Forrester 1961) to model the courseware development with the aim of better understanding the process. This research is intended to provide a simulation environment in which courseware project managers can explore various aspects,

including *effort*, *development time* and risk, which will help them to understand the underlying relationships that exist within courseware development projects. As such it attempts to replicate work carried out in the area of software engineering metrics using system dynamic models (Abdel-Hamid and Madnick 1989; Abdel-Hamid 1990; Abdel-Hamid and Madnick 1991; Abdel-Hamid 1993; Abdel-Hamid et al. 1993). However, at present there is no published research on the use of the model to estimate effort. In 1995 only the analyst sub-system of the courseware model had been developed and it had not been validated against actual project data, only expert opinion (Davidsen et al. 1995). Spector (1996) stated there are plans to try to validate the model against actual large scale projects in the future, but felt that these may not come to anything due to the difficulty in collecting data. While recognising the problem, the model will be of less value if it continues to be based on just the opinions of a small group of experts.

2.3 EVALUATION OF EXISTING METHODS

One of the main problems associated with multimedia courseware effort estimation is that there has been no critical evaluation of the existing methods. Any evaluation is usually based on the opinion of the author, or, at best, the opinion of a small group of experts. Typically, there is no evaluation of the method's predictive accuracy and consistency against actual project data. In one study which did attempt to measure the predictive accuracy of the method, the full results were not published and the data set may well have been used in the original construction of the method (Schooley 1988). The only published independent evaluation of courseware effort estimation methods was carried out by Jay, Bernstein and Gunderson (1987) for the US Army. In their evaluation of Kearsley's (1985) CBT Analyst they found one software bug and a number of sizing factor ratings which required clarification by the author before the method could be used with an independent data set.

Despite the previously reported error in their analysis of estimates produced by CBT Analyst, Jay, Bernstein and Gunderson (1987) did find a number of weaknesses and made some recommendations on ways to improve the method (Jay et al. 1987; Jay 1988). Unfortunately, this informal study did not look at the whole picture and perhaps focused too readily on evaluating the "ease of use" of the method and the "fidelity" of the estimates produced.. These are only two of ten criteria identified by Boehm and Wolverton (1980) to help evaluate the utility of a software effort estimation method for practical estimation purposes. Table 2.27 shows the Boehm and Wolverton (1980) criteria for evaluating software effort estimation methods refocused to deal with courseware effort estimation methods.

Table 2.27 - Criteria for evaluating courseware *effort* methods (Boehm and Wolverton 1980)

Criterion	Description
1. Definition	Has the model clearly defined the [effort] it is estimating, and [effort] it is excluding?
2. Fidelity	Are the estimates close to the actual [effort] expended on the projects?
3. Objectivity	Does the model avoid allocating most of the [courseware effort] variance to poorly calibrated subjective factors (such as complexity), i.e. is it hard to rig the model to get the results you want?
4. Constructiveness	Can you tell a user why the model gives the estimate that it does? Does it help the user understand the [courseware development] to be done?
5. Detail	Does the model easily accommodate the estimation of a [courseware] system consisting of a number of subsystems and units? Does it give (accurate) phase and activity breakdown?
6. Stability	Do small differences in inputs produce small differences in output [effort] estimates?
7. Scope	Does the model cover the class of [courseware] projects whose [effort] you need to estimate?
8. Ease of use	Are the model inputs and options easy to understand and specify?
9. Prospectiveness	Does the model avoid the use of information which will not be known until the project is completed?
10. Parsimony	Does the model avoid the use of highly redundant factors or factors which make no appreciable contribution to the result?

These ten criteria help to highlight some of the problems associated with existing courseware effort estimation methods. In this Section the non-algorithmic and algorithmic methods previously identified will be briefly evaluated using the ten criteria in Table 2.27. The methods which are solely based on expert opinion will only be evaluated where they provide a useful comparison with the other two classes of methods.

2.3.1 Definition

The definition criterion evaluates what is included or excluded from the development effort being estimated. Evaluation of the methods identified in this Chapter indicates that all fail the definition criterion. None of the methods clearly and unambiguously define the effort they are including or excluding from the estimate. As a minimum the methods would have to define a measurement baseline consisting of:

- Organisational context
- Courseware quality
- Development life-cycle
- Development activities

It is essential to define the organisational context for which a method was originally developed. A method developed originally by a sole author working in an educational

environment producing prototype classroom aids is unlikely to be useful to a manager working with a large team for a commercial client. The existing methods do not specify the intended organisational context for the method or the source of the data used to develop the method. Similarly, the expected courseware quality has to be clearly defined because the quality required by a boxed-set of multimedia courseware for public release may have very little in common with quality required for a partially finished classroom prototype.

There is also no consensus on which phases are included in the courseware development life-cycle used to measure the effort expended. Those methods which actually define the life-cycle they are using normally do not agree on the start and end points for measuring effort. For example, Schooley (1988) specifically excludes training needs analysis which Bergman and Moore (1990) specifically include. The only general agreement is that they are not concerned with the maintenance phase but none of them define a way to differentiate between the end of the testing and the start of the maintenance phases.

As Jay, Bernstein and Gunderson (1987) indicated it is difficult to evaluate courseware effort estimation methods if there is no indication of the development activities included in the estimation. For example, Golas (1993) included project management effort "...of approximately ten percent ... in the baseline estimate", but what about the other activities? Does the estimate include or exclude administrative and secretarial support, video production or the Subject Matter Expert's effort? None of the methods reviewed clearly defined the development activities which were included in their estimates of effort. The tendency was to define the included development activities in very general terms or, more commonly, not to define them at all.

2.3.2 Fidelity

The fidelity criterion asks how close to the actual effort expended on the projects are the estimates produced by the method? This is the basic question most courseware project managers would ask. Boehm and Wolverton (1980) in their discussion on fidelity considered it only as one criterion. However, it can be separated into two related classes which are called:

- Internal fidelity
- External fidelity

Internal fidelity evaluates the predictive accuracy of the estimation method when applied to the type of project for which it was originally developed using the data set from which the model was originally developed. External fidelity looks at the predictive accuracy of the method when used with a calibrated data set from the type of projects for which the method was originally developed. Both classes of fidelity can be measured using $\overline{MRE}$ with *PRED(l)* and the effort estimation evaluation scheme (Campbell et al. 1988).

Unfortunately, because none of the methods are well defined it is difficult to evaluate internal fidelity. The only authors who published data on internal fidelity were Schooley (1988) and Marshall et al. (1994c). Unfortunately, Schooley (1988) did not define the type of courseware project the method was designed to estimate and did not directly report $\overline{MRE}$ with *PRED(l)*. Table 2.28 shows the $\overline{MRE}$ and *PRED(l)* reported by effort estimation methods in comparison with those produced by expert estimation.

Table 2.28 - Comparison of expert estimation and published effort estimation method results

Estimation method	$\overline{MRE}$	PRED(0.25)	PRED(0.30)
Baseline estimate (*BE*)	0.51	0.18	0.18
Project complexity multipliers (*PCM*)	0.57	0.45	0.55
Multimedia effort estimation method (*MEEM*)	0.27	0.60	0.70
Cost estimating algorithm for courseware (*CEAC*)	0.31	0.58	0.58
CBT analyst evaluation - Mean	0.93	0.22	0.22

Three of the methods produced values for $\overline{MRE}$ and *PRED(l)* which fall in the poor method range and this indicates poor internal fidelity. Only MEEM just manages to reach the acceptable method criterion for $\overline{MRE}$ and *PRED(l)*. Similarly, in evaluating external fidelity, unless the method is calibrated to the project data and the data is relevant to the type of courseware for which the method was developed, then the outcome will be very poor external fidelity. This can be seen in the results of Jay, Bernstein and Gunderson's (1987) informal evaluation of CBT Analyst. As Table 2.28 shows, the $\overline{MRE}$ and *PRED(l)* obtained using data from nine uncalibrated independent projects produces very poor external fidelity results.

Marshall Samson and Dugard (1994a) using a well defined MEEM (Marshall et al. 1994c) data set from ten projects evaluated three of the other effort estimation methods described in this Chapter. Table 2.29 shows the relative errors for the estimates produced by the USA Airforce interactive courseware method (Golas 1993), CEAC (Schooley 1988) and CBT Analyst (Kearsley 1985). The three methods were selected on the basis that they appeared to match the types of courseware and development environment which had existed in the production of the ten projects for which the data was available. As previously mentioned, this is not defined directly by any of the methods. The selection of the three methods was made informally using indications of the method author's organisation, production methods and technology gleaned from the supporting literature. It can only be viewed as a very crude attempt to match the method to the external data set.

Table 2.29 - Relative errors of estimate for the four estimation methods (Marshall et al. 1994a)

Project	MEEM	USA ICW	CEAC	CBT Analyst		
				Low	Mean	High
1	0.63	1.06	3.36	0.25	0.88	1.50
2	0.89	3.20	2.19	1.00	2.00	3.00
3	0.00	0.50	0.74	0.00	0.50	1.00
4	0.29	0.53	0.24	0.44	0.17	0.11
5	0.25	1.10	3.40	0.00	0.50	1.00
6	0.41	0.25	0.22	0.55	0.32	0.09
7	0.11	0.52	0.24	0.20	0.20	0.60
8	0.03	0.63	0.46	0.38	0.06	0.25
9	0.07	0.05	0.20	0.50	0.25	0.00
10	0.02	0.18	5.40	0.00	0.50	1.00

As can be see from Table 2.29, some of the estimates produced extremely large relative errors. This is reflected in the range of $\overline{MRE}$ and *PRED(l)* values shown in Table 2.30.

Table 2.30 - Evaluation of estimation method external fidelity

	MEEM	USA ICW	CEAC	CBT Analyst		
				Low	Mean	High
$\overline{MRE}$	0.27	0.80	1.64	0.33	0.54	0.86
PRED(0.20)	0.50	0.20	0.00	0.40	0.40	0.30
PRED(0.25)	0.60	0.30	0.40	0.50	0.50	0.50
PRED(0.30)	0.70	0.30	0.40	0.50	0.50	0.50

While calibration would undoubtedly improve these results, this is difficult if there is no evidence of the original data set on which to base the calibration. Even roughly matching the method to the external data set did not produce acceptable results for

external fidelity.

2.3.3 Objectivity

How difficult is it to rig the methods to get the results you want? In terms of existing courseware effort estimation methods the answer to this question must be "very easy". The methods tended to use poorly calibrated subjective factors such as complexity as the basis for effort estimation. Jay, Bernstein and Gunderson (1987) in their evaluation of CBT Analyst required considerable clarification by the method's developer of a number of the factors used. For example, one of the questions requires the estimator to rate the experience level of the developer. The original factors consisted of "very experienced", "some experience" or "no experience". Kearsley (1987) in a personal communication quantified these factors as "5 or more years", "1 to 5 years" and "less than one year" for the evaluation. This is only one example, other methods abound with poorly calibrated subjective descriptions which are used to tailor the estimate of effort.

However, the major failing of all the methods described is the absence of a clearly defined base size metric. Nine out of the ten methods used *learner time* as the base size metric for estimation. Fairweather and O'Neal (1984) described *learner time* as "the most slippery metric known to man". It is not clearly defined how it is measured or estimated with the result that choosing the appropriate *learner time* can produce very good estimates of effort. This reinforces the need to define what is being measured and how it is to be measured.

2.3.4 Constructiveness

Boehm and Wolverton (1980) in discussing this criterion provided examples of tables which clarified for a developer the effect of different ratings and factors on the project. In these terms none of the methods are constructive in that most are informally defined with little or no help. In addition. the direct effect of changing a rating on the overall estimate is not obvious. That is not to say that the methods could not be improved to provide additional help and support which would improve the consistency in rating and explain the effect on overall effort.

The other aspect of this criterion is the question "Does it help the user understand the

[courseware development] to be done?" The answer to this question is probably "Yes". Despite all the problems with the methods they do encourage the developer to review the courseware specification and then consider a number of factors which may affect the project to be undertaken. All the available evidence on the use of courseware effort estimation models points to this conclusion (Jay et al. 1987; Golas 1993).

2.3.5 Detail

The detail criterion looks at the method's ability to deal with estimates using a bottom up approach. Only the "quick and dirty" method (Miles 1990a; Miles and Griffith 1993) is designed to produce estimates from the consideration of the individual life-cycle phases. There is some evidence that the method may give accurate phase breakdown because it is similar to the other percentage estimates of phase distribution (Sampath and Quine 1988). However, this has not been evaluated using actual project data.

All the other methods base their estimates around questions related to overall development with little, if any, reference to the individual phases of the life-cycle. Those methods which informally define a life-cycle do not use the same definition for the number of phases or their names and this makes it impossible to compare the results.

2.3.6 Stability

Boehm and Wolverton (1980) defined a stable method as one which for small differences in inputs produced small differences in effort estimates. This aspect of stability can be measured using the range of *effort to learner time ratios* produced by the method and the number of steps used to produce this range. However, in their discussion Boehm and Wolverton (1980) also presented an example of an unstable model which had neighbourhoods of severe discontinuity for certain input values. So, in assessing the stability of a method, any regions where there are unpredictable changes in output for small changes in input have to be identified. To assess the stability of a method, the percentage of the mean step size over the range of *effort to learner time ratio* is calculated. Table 2.31 shows the mean step stability of the courseware estimation methods and the expert opinion-based industry average for comparison.

Table 2.31 - Stability of courseware estimation methods

Method	Range	Number of steps	Mean step size	Mean step stability (%)
Courseware cost estimating algorithm[18]	57596.0	196000	0.29	0.0005
Industry averages[19]	3999.0	399.9	10.00	0.25
Regression method[20]	203.4	240	0.85	0.42
Multimedia effort estimation method[21]	2046.1	237	8.63	0.42
Quick and dirty[22]	420.0	56	7.50	1.79
Q-factor analysis[23]	2450.0	50	49.00	2.00
Air force interactive courseware method[24]	1360.0	24	56.67	4.17
Training cost calculation spreadsheet[25]	360.0	22	16.36	4.55
CBT Analyst[26]	650.0	4	162.50	25.00
Computer-based instruction length method[27]	162.0	3	54.00	33.33
Estimation grids[28]	332.5	3	110.83	33.33

Table 2.31 shows the range of step size stability percentage produced by the methods. CEAC is the most stable with a mean step size of 0.29 $Dh\ \overline{Lh^{-1}}$ or 0.0005% of the range of *effort to learner time ratios*. Small changes in this method's sizing factors do produce small changes in the *effort to learner time ratios* over the large range of output values it produces and there are no regions of discontinuity. The least stable method is Estimation grids which have large mean step size as a percentage of the range of *effort to learner time ratio*. This method also has discontinuities at critical points where small changes in input can produce very large changes in *the effort to learner time ratio*. The discontinuity is also unpredictable in that changing the values of a large number of the sizing factors may produce no effect then suddenly the mean *effort to learner time ratio* almost doubles.

Given that the aim is to produce a method which is capable of producing estimates with a $\overline{MRE}$ consistently less than 0.25, any method with a mean step size stability of 25% or greater is unlikely to produce an acceptable result. A much smaller mean step size stability percentage would be required to meet this target. Using the criterion that the mean step size stability should be less than or equal to 2.5%, only six of the methods in Table 2.31 meet the criterion.

2.3.7 Scope

As indicated previously, none of the methods clearly define their scope. All that can be said for most of the methods is that they estimate courseware development effort.

Limitations in terms of project size, delivery techniques, team size and development methods are seldom specified directly. As can be seen from Table 2.32, the range of *effort to learner time ratio* estimates produced by the methods varies considerably.

Table 2.32 - Range of *effort to learner time ratios* produced by each method

Method	Minimum	Maxmum
Industry averages	1	4000
Multimedia effort estimation method	1.9	2048
Courseware cost estimating algorithm	4	57600
Computer-based instruction length method	18	180
Air force interactive courseware method	30	1390
Training cost calculation spreadsheet	40	400
Regression method	40.2	243.6
Q-factor analysis	50	2500
Estimation grids	85-150	300+
CBT Analyst	<100	500+
Quick and dirty	171	591

It is hard to believe from the range of values shown in Table 2.32 that these methods could possibly have the same scope. They most obviously do not but the authors have not specified clearly the limitations of their method. While it is possible to wade through the documentation and make assumptions about the scope from the method factors, it would be far simpler if the scope was clearly specified at least in terms of:

- Courseware delivery techniques
- Authoring tools
- Media range
- Team size range
- Project elapsed time range
- Delivery platform
- Intended learner population
- Organisational context
- Life-cycle starting and end points
- Courseware product quality

Without these being clearly specified it is difficult to match the method to the courseware to be developed or to calibrate the method to the new circumstances.

2.3.8 Ease of use

All the methods are easy to use and the calculations can be automated using a spreadsheet (Marshall 1995). However, Boehm and Wolverton's (1980) ease of use criterion looked beyond its most obvious meaning to evaluate how easy the inputs and options used by the method were to understand and specify. The input is particularly difficult to understand with ample scope for different interpretations by estimators rating the same project. This specific problem was reported by Kearsley (1987) who, in an informal validation of CBT Analyst, found that five courseware experts were unable to reach a consensus on a single answer to any question for five separate courseware specifications.

It is perhaps wrong to keep picking out CBT Analyst just because it was used in Jay, Bernstein and Gunderson's (1987) study. The other methods share its weaknesses by using sizing factors based on vague questions. For example, what is the answer to this Air force interactive courseware method (Golas 1993) size factors statement?

> "Inadequate documentation. No training needs assessment was performed. No task analysis or learning analysis data. Technical manuals are non-existent or are not helpful"

Do you increase the *effort to learner time ratio* if all five criteria are true, or if only one is true? How is the inadequacy of the documentation to be judged? What happens if a task analysis was not performed but the technical manuals are available and are helpful? These are the types of question that would have to be clarified before there would be any hope of producing consistent results using the size factor question in the current courseware effort estimation methods. Even when the methods use less subjective questions or statements as the basis for allocating a size factor, the rating scales can be inconsistent or contradictory.

In their original format the methods studied output estimates in a range of units which normally confused *development time*, *effort*, *cost* and *effort to learner time ratios* using an equally diverse range of units. This rationalisation of the methods has helped to clarify the situation and make clear what is being estimated. This improves their ease of

use.

2.3.9 Prospectiveness

A potential problem with effort estimation methods is that they can depend on information which will not be accurately known until later in the courseware development life-cycle or even until the project is completed. However, effort estimation methods are most useful at the start of the development when reliable information does not exist. The prospectiveness criterion evaluates the extent to which the method avoids using information which will not be known until the project is completed. As previously indicated, nine out of the ten existing methods make use of *learner time* as the base size metric. This can only be measured once the development project has been completed. Up to that point it can only be estimated by the developer or specified by the client.

In general, the methods reviewed do not make use of sizing factors which will only be known once the project is finished. The exception to this rule is the Air force interactive courseware method (Golas 1993) which requires the estimator to decide if the customer is using consistent acceptance criteria.

> "Customer is not using objective and consistent acceptance criteria. Customer is unsure of what is wanted and does not communicate with developer."

Since acceptance criteria are not normally used until the project is under way, it is difficult to know if the "customer is not using objective and consistent acceptance criteria". The question could be re-framed to rate the likelihood that the client will use objective and consistent acceptance criteria. However, all of the methods used size factors which can only be known or estimated after the courseware is reasonably well specified.

2.3.10 Parsimony

The parsimony criterion evaluates how well the methods avoid the use of highly redundant sizing factors which make no appreciable contribution to the final result.

Most of the methods suffer from the use of redundant or overlapping sizing factors. For example, Gery's (1987) Estimation grids make use of the following five sizing factors:

- Nature and frequency of interactivity
- Conditional branching
- Response analysis: complexity, feedback and branching
- Nature and depth of feedback
- Nature and depth of testing

These all appear to measure the interactivity and are at least partially dependent on each other. These five sizing factors could be replaced by one sizing factor without changing the final estimate. Since this method uses thirty-seven sizing factors to produce just three alternative ranges of *effort to learner time ratios,* reducing the number will have very little effect on the accuracy of the estimate. At the other extreme is the regression method (Senbetta 1991) which requires only total *learner time* to produce an estimate. While it is parsimonious, it provides no opportunity to tailor the estimate to the courseware or development environment using sizing factors. All the other methods lie somewhere between these two extremes using between five and twenty-four sizing factors to tailor the final estimate. There has been no published attempt to establish the independence of these sizing factors or the effect of each sizing factor on the overall estimate.

2.4 SUMMARY

One of the problems associated with courseware effort estimation is the confusing array of terms and measurements used by different authors. In this Chapter, an initial measurement paradigm has been established to enable meaningful conclusions to be drawn from the bewildering range of terms used to describe effort estimation methods in the literature. This rationalisation, along with the use of $\overline{MRE}$ and *PRED(l)* criteria proposed by Campbell, Conte and Rathi (1988), has established a method for evaluating courseware effort estimation methods.

Five basic types of expert opinion-based methods for estimating multimedia courseware effort were identified from the literature. The most widely quoted method is through the use of industry average *effort to learner time ratios*. There is such a wide range of values quoted for industry averages that they are useless except as a starting point for further refinement. There is no evidence of any systematic attempt to collect and evaluate the accuracy of estimates based on these averages. However, experience of work breakdown with and without factors to tailor the initial estimate to the circumstances in which the courseware was to be developed proved disappointing. These studies produced results which were consistently in the poor method classification ($\overline{MRE} > 0.30$ and $PRED(0.30) < 0.70$).

A further eleven courseware effort estimation methods were identified as either algorithmic or non-algorithmic. These eleven methods were promoted by their respective authors as potentially providing more reliable results than expert opinion. However, there is little evidence to indicate that any of these methods have been internally or externally validated using independent data. The only exceptions are Schooley (1988) and Marshall et al. (1994c) who produced estimates which were in the poor and adequate method classification respectively. Boehm and Wolverton's (1980) ten criteria for evaluation effort estimation models highlighted the problems with both the algorithmic and non-algorithmic methods. The major problem is the lack of consistent definitions which makes it difficult to know what was being estimated. This in turn makes the collection and measurement of courseware development data to allow calibration and validation of potential models almost impossible.

Throughout this Chapter the techniques found in the literature have been described consistently as methods not models. The choice of term is based on analysis of the evaluation criteria and a judgement to the extent to which rigorous techniques were used to construct and validate the method. The methods presented are poorly defined with little evidence of validation. As such, they have more in common with the expert opinion techniques than a rigorous model which predicts effort. The lack of a measurement paradigm and standard techniques for data collection, construction, validation and evaluation of models severely hampers research in to courseware effort estimation.

None of the present courseware effort estimation methods meet the minimum criteria for consideration as a good model for courseware effort estimation. In the next Chapter, a detailed description of the measurement paradigm will be presented as the basis for collecting data to construct and validate courseware development effort estimation models.

2.5 REFERENCES

Abdel-Hamid, T.K., 1990. On the utility of historical project statistics for cost and schedule estimation: results from a simulation-based case study. *Journal of Systems Software*. **13**: pp. 71-82.

Abdel-Hamid, T.K., 1993. A multiproject perspective of single-project dynamics. *Journal of Systems Software*. **22**: pp. 151-165.

Abdel-Hamid, T.K. and Madnick, S.E., 1989. Lessons learned from the dynamics of software development. *Communications of the ACM*. **32(12)**: pp. 1426-1438.

Abdel-Hamid, T.K. and Madnick, S.E., 1991. *Software project dynamics: An integrated approach*. Englewood Cliffs, NJ: Prentice-Hall.

Abdel-Hamid, T.K., Sengupta, K. and Ronan, D., 1993. Software project control: An experimental investigation of judgement with fallible information. *IEEE Transactions on Software Engineering*. **19(6)**: pp. 603-612.

Avner, A., 1988. Is there an ideal size for courseware production teams? In: *30th Association for the Development of Computer-Based Instructional Systems, Philadelphia, 7-10 November*. pp. 143-147.

Avner, A., Smith, S. and Tenczar, P., 1984. CBI authoring tools: effects on productivity and quality. *Journal of Computer-Based Instruction*. **11**: pp. 85-89.

Avner, R.A., 1979. Production of computer-based instructional materials. In: H.F. O'Neil, Editor *Issues in instructional systems development*. New York: Academic Press, pp. 133-180.

Baker, J., 1994. One man and his dog. *Interact*. **1(3)**: pp. 16-17.

Barron, A. and Fisher, H., 1993. Affordable videodisc production: a model for success. *Tech Trends*. **March**: pp. 15-21.

Beautement, P., 1991. Review of interactive video systems and their possible application to training in the 90's. *Interactive Learning International*. **7**: pp. 45-54.

Bergman, R.E. and Moore, T.V., 1990. *Managing interactive video/multimedia projects*. Englewood Cliffs, NJ: Educational Technology Publications.

Boehm, B.W., 1981. *Software engineering economics*. Englewood Cliffs, NJ: Prentice-Hall.

Boehm, B.W. and Wolverton, R.W., 1980. Software cost modelling: Some lessons learned. *Journal of Systems and Software*. **1(3)**: pp. 195-201.

Bourdeau, J. et al., 1995. Automating instructional planning. In: R.D. Tennyson and A.E. Barron, Editors *Automating instructional design: computer-based development and delivery*. Berlin: Springer Verlag, pp. 559-569.

Bunzel, M.J. and Morris, S.K., 1992. *Multimedia application development: using DVI technology*. New York: McGraw-Hill.

Campbell, R.L., Conte, S.D. and Rathi, M.K., 1988. *Early prediction of software size and effort*. Technical Report (SERC-TR-10-P), Purdue University.

Casey, R.J. et al., 1988. Capturing skill in estimating training development effort: A follow-up final report. *Performance & Instruction*. **27(9)**: pp. 40-44.

Conte, S.D., Shen, V.Y. and Dunsmore, H.E., 1986. *Software engineering metrics and models*. Menlo Park: Benjamin/Cummings Publishing Company Inc.

Davidsen, P. et al., 1995. Applying system dynamics to courseware development. *Computers in Human Behavior*. **11(2)**: pp. 325-339.

De Diana, I. and van Schaik, P., 1993. Courseware engineering outlined: An overview of some research issues. *Education & Training Technology International*. **30(3)**: pp. 191-211.

Dean, C. and Whitlock, Q., 1983. *A handbook of computer-based training*. London: Kogan Page.

Diaper, D., Editor 1989. *Knowledge elicitation: Principles, techniques and applications*. New York: Ellis Horwood.

Fairweather, P. and O'Neal, A., 1984. The impact of advanced authoring systems on CAI productivity. *Journal of Computer-based Instruction.* **11**: pp. 90-94.

Fenton, N., Editor 1991. *Software metrics: a rigorous approach.* London: Chapman-Hall.

Fenton, N., 1994. Software measurement: a necessary scientific basis. *IEEE Transactions on Software Engineering.* **20(3)**: pp. 199-206.

Forrester, J.W., 1961. *Industrial dynamics.* Cambridge, MA: MIT Press.

Fox, J., 1991. *When worlds collide: demystifying multimedia.* PC Today. **June: p. 6.**

Galbreath, J., 1992. The coming of digital desktop media. *Educational Technology.* **June**: pp. 27-32.

Gayeski, D.M., 1992. Making sense of multimedia: Introduction to special issue. *Educational Technology.* **May**: pp. 9-13.

Gerard, R.N., 1967. *Computers and education.* New York: McGraw-Hill Book Company.

Gery, G., 1987. *Making CBT happen: Prescriptions for successful implementation of computer-based training in your organisation.* Boston, MA: Weingarten Publications Inc.

Golas, K., 1994. *Personal communication.* (E-mail to I. Marshall, 28 January 1994) SWRI.

Golas, K.C., 1993. Estimating time to develop interactive courseware in the 1990s. In: *13th Interservices Industry Training and Education Conference, Orlando, FL, November 1990.*

Goodyear, P., 1995. Infrastructure for courseware engineering. In: R.D. Tennyson and A.E. Barron, Editors *Automating instructional design: computer-based development and delivery tools.* Berlin: Springer-Verlag, pp. 11-31.

Hannafin, M. and Peck, K., 1988. *The design, development and evaluation of instructional software*. New York: Macmillan.

Heemstra, F.J., 1990. Software cost estimation model. In: *5th Jerusalem Conference on Information Technology, Jerusalem, Israel,* pp. 286-297.

Heemstra, F.J., 1992. Software cost estimation. *Information and Software Technology.* **14(10)**: pp. 627-639.

Hurlock, R.E. and Slough, D.A., 1976. *Experimental evaluation of PLATO IV technology: Final report.* Final Report (NPRDC TR 76 TQ-44 (AD A029 384)), August 1976, San Diago, CA: Navy Personnel Research and Development Center.

Imke, S., 1991. *Interactive video management and production.* Englewood Cliffs, NJ: Educational Technology Publications.

Jay, J., 1988. The CBT analyst. *Journal of Computer-Based Instruction.* **15(1)**: pp. 34-45.

Jay, J., Bernstein, K. and Gunderson, S., 1987. *Estimating computer-based training development times.* ARI Technical Report (765), October 1987, Alexandria, VA: Research Institute for Behavioural and Social Sciences.

Jones, M.K., Zhongmin, L. and Merrill, D.M., 1993. Rapid prototyping in automated instructional design. *Educational Technology Research & Development.* **40(4)**: pp. 95-100.

Kearsley, G., 1985. The CBT advisor: An expert system program for making decisions about CBT. *Performance and Instruction.* **24(9)**: pp. 15-17.

Kearsley, G.P., 1987. *Personal communication.* (Letters to J. Jay, May-June 1987) Washington: George Washington University. Quoted in: Jay, J., Bernstein, K. and Gunderson, S., 1987. *Estimating computer-based training development times.* ARI Technical Report (765), October 1987, Alexandria, VA: Research Institute for Behavioural and Social Sciences.

Kitchenham, B., Pfleeger, S.L. and Fenton, N., 1995. Towards a framework for software measurement validation. *IEEE Transactions on Software Engineering*. **21(12)**: pp. 929-944.

Kulik, C.C. and Kulik, J.A., 1986. Effects of computer-based education in colleges. *Association for Educational Data Systems Journal*. **19**: pp. 81-108.

Lee, C. and Zemke, R., 1987. How long does it take? *Training*. **24(6)**: pp. 75-80.

Lisewski, B. and Settle, C., 1995. Teaching with multimedia: a case study in weed biology. *Active Learning*. **3(December)**: pp. 28-35.

Lokan, C.J., 1996. Early size prediction for C and Pascal programs. *Journal of Systems Software*. **32(1)**: pp. 65-72.

Marshall, I.M., 1995. *Multimedia courseware effort and cost models*. Dundee: Marshall Associates.

Marshall, I.M., Samson, W.B. and Dugard, P.I., 1994a. Multimedia courseware. Never mind the quality, how much will it cost to develop? In: *Association for Learning Technology 94, University of Hull, Hull, 18-21 September 1994*.

Marshall, I.M., Samson, W.B. and Dugard, P.I., 1994b. Predicting the development effort and cost of multimedia courseware for open access education. In: *29th International Conference of the Association for Educational Technology, Napier University, Edinburgh, 11-13 April*.

Marshall, I.M. et al., 1995. The mythical development to learner time ratio. *Computers and Education*. **25(3)**: pp. 113-122.

Marshall, I.M. et al., 1994c. Predicting the development effort of multimedia courseware. *Information and Software Technology*. **36(5)**: pp. 251-258.

Mikos, R. et al., 1987a. Estimating training effort/cost: A follow-up 'final report' (Part 2). *Performance & Instruction*. **26(7)**: pp. 33-43.

Mikos, R. et al., 1988. Only experienced estimators need apply. *Performance & Instruction.* **27(2)**: pp. 56-60.

Mikos, R. et al., 1987b. Estimating training effort/cost: A follow-up 'final report' (Part 1). *Performance & Instruction.* **26(5)**: pp. 24-29.

Miles, K., 1990a. *CBT product definition of the C-17 aircrew training system.* Unpublished Manuscript Naperville, IL: National Educational Training Group.

Miles, K., 1990b. A generic model for rapid estimation of CBT development time. In: *13th InterAgency/Industry Training Systems Conference, Orlando, FL, November 1990.*

Miles, K.W. and Griffith, E.R., 1993. Developing an hour of CBT: The quick and dirty estimate. *CBT Directions.* **(April-May)**: pp. 28-33.

Milette, M. and Trevor-Deutsch, L., 1995. *Training cost calculation spreadsheet: classroom versus computer-based training.* Visual Solutions Inc.

O'Neil, H.J., 1987. *Personal communication.* (Letter to J. Jay, October 1987) . Quoted in: Jay, J., Bernstein, K. and Gunderson, S., 1987. *Estimating computer-based training development times.* ARI Technical Report (765), October 1987, Alexandria, VA: Research Institute for Behavioural and Social Sciences.

Orey, M. et al., 1992. High bandwidth diagnosis within the framework of a microcomputer-based intelligent tutoring system. *Journal of Artificial Intelligence in Education.* **3(1)**: pp. 63-80.

Orey, M. et al., 1994. Development efficiency and effectiveness of alternative platforms for intelligent tutoring for the mobile subscriber radio-telephone terminal. *Computers and Education.* **22(4)**: pp. 301-313.

Orlansky, J. and String, J., 1979. *Cost-effectiveness of computer-based instruction in military training.* Technical Report (P-1375), April 1979, Institute for Defence Analysis.

Parker, T., 1983. *Rules of thumb.* Boston, MA: Houghton Mifflin Co.

Philips Professional Media, 1996. *Costing a production.* UK: Philips Media Professional.

Ralston, G., 1991. Hypermedia ... not multimedia. *The expanded desktop.* **1(4)**: p. 58. Quoted in: Galbreath, J., 1992. The educational buzzword of the 1990's: Multimedia, or is it hypermedia, or is it interactive video, or ...? *Educational Technology.* **April**: pp. 16-19. .

Sampath, S. and Quine, A., 1988. Effective interface tools for CAI authors. *Journal of Computer-based Instruction.* **17(1)**: pp. 31-34. Quoted in: Miles, K., 1990. A generic model for rapid estimation of CBT development time. In: *13th InterAgency/Industry Training Systems Conference, Orlando, FL, November 1990.*

Schooley, R.E., 1988. Computer-based training (CBT) cost estimating algorithm for courseware (CEAC). In: *11th Interservice Industry Training Systems Conference,* pp. 319-328.

Senbetta, G., 1991. *An inquiry of time and cost estimating for computer-based training courseware design and development as determined by modified delphi method.* (Ph.D. thesis, Purdue University).

Senbetta, G., 1992. CBT time and cost estimation: what do the experts say? In: *10th Annual CBT Conference and Exposition,*

Soloman, M.B., 1994. What's wrong with multimedia in higher education? *Technological Horizons in Education Journal.* **21(7)**: pp. 81-83.

Spector, M., 1996. *Personal communication.* (e-mail to I.M. Marshall) Bergen, Norway: University of Bergen.

Stack, P., 1990. Interactive video - the barriers have fallen. *Tech Trends.* **35(2)**: pp. 38-40.

Stewart, A.M. and Bryce, C.F.A., 1981. Multimedia multipurpose: Is the quality of the learning experience being well served by the use of educational media? In: F. Percival and H. Ellington, Editors *Distance learning and education.* London: Kogan Page,

Symons, C.R., 1993. *Software sizing and estimating: Mk II FPA (function point analysis)*. Chichester: John Wiley & Sons.

Tan, W. and Nguyen, A., 1993. *Lifecycle costing models for interactive multimedia systems: Interactive multimedia practice and promise*. London: Kogan Page.

Tennyson, R.D. et al., 1995. Employment of system dynamics in modelling of instructional design (ISD). In: R.D. Tennyson and A.E. Barron, Editors *Automating instructional design: computer-based development and delivery tools*. Berlin: Springer-Verlag, pp. 603-609.

US Navy, 1987. *Request for technical proposal*. RTP (Navy Contract N61339-87-R-2043), US Navy. Quoted in:Miles, K., 1990. A generic model for rapid estimation of CBT development time. In: *13th InterAgency/Industry Training Systems Conference, Orlando, FL, November 1990.*

Woolf, E., Tanna, S. and Singh, K., 1985. *Quantitative analysis*. Estover, Plymouth: Macdonald & Evans Ltd.

2.6 ENDNOTES

[1] Mikos et al. (1987b) used "time" measured in hours as the basis for their estimation experiments. The paradigm definition of effort is substituted.

[2] Jay, Bernstein and Gunderson (1987) used "cost" in several ways throughout the document to describe both effort and monetary value. In this particular reference the survey respondents were asked to estimate the accuracy of their *cost* estimates. The paradigm definition of *cost* is therefore used.

[3] Senbetta (1991) used "Level of effort estimation for each hour of computer-based training" which he measured in "hours per hour". The paradigm definition of *effort to learner time* ratio is substituted.

[4] Mikos et al. (1987b) used "time" measured in hours as the basis for their estimation experiments. However, they repeatedly used "cost", "workload" and "effort" to describe what they were measuring. The paradigm definition of *effort* is substituted.

[5] Imke (1991) used "time" measured in "hours" to describe the *effort* expended by individual members of the project team. The result is converted to *cost* in "$" using a "Charge rate" measured in "$". To ensure consistency with the paradigm, the "Charge rate" is converted to *$ per developer-hour ($ Dh^{-1})*.

[6] Philips Professional Media (1996) only considered the cost of producing multimedia titles using monetary value in pounds. The paradigm definition of *cost* is therefore used.

[7] "Industry average" is reported in a wide range of different units - "hour per hour", "developer time per student hour" and "average developer time per hour" being typical. They are usually described as "X:1 hours per student hour" or some other variation. The base size estimate is normally described in terms of "student-hours" or some other variation on this theme. The resulting estimate is typically described as

"time" required to develop the courseware measured in "hours". The paradigm definitions of *effort to learner time*, *learner time* and *effort* are substituted.

[8] Gery (1987) claimed to be estimating development "time" and "cost" but the output of the model is three ranges of "development ratio" measured in "development hours per CBT hour". The units have been converted to match those of the paradigm and the output is *effort to learner time ratio*.

[9] Golas's (1993) original description was of a method which could estimate "time" to develop interactive courseware. The method produced estimates of the "hours needed to develop one hour of ICW". This has been converted to *effort to learner time ratio* measured *in mean-developer-hours-per-learner-hour* which, with the addition of an estimate for *learner time*, will estimate *effort*.

[10] Bergman and Moore (1990) only provided a method of estimating courseware cost measured in "$" based on the required "delivery time" measured in "hours". The paradigm definition of *cost* is therefore used and the *Q-factor* method extended to allow comparison with other methods.

[11] Tan and Nguyen (1993) were primarily concerned with total life-cycle "cost" measured in Australian Dollars (A$). They estimated the *learner time* from the instructor-led equivalent course unit length measured in hours. The paradigm definitions of *learner time*, *cost*, *effort* and *effort to learner time ratio* are substituted.

[12] Marshall et al. (1994c) use "effort" and "development time" interchangeably as well as using "delivery time". The paradigm definition of *learner time* and *effort* are substituted.

[13] Kearsley's (1985) model estimated the development "cost" but produced estimates of "development hours per instructional hour". The paradigm definition of *effort to learner time ratio* is substituted.

[14] Schooley (1988) based his model on initially estimating "development time" measured in "hours" which was then converted into "effort" measured in "staff-months" which was finally converted in to "cost" measured in "$". The model has been converted into the paradigm terms with *effort* used in place of "development time". The other outputs of the model have been mapped to measure *cost* and *effort* using measures based on *developer hours*.

[15] Miles and Griffith's (1993) original model used "development time" measured in "hours". However there is an unstated assumption that the model is based on one hour of learner time. The paradigm definition of *learner time*, *effort* and *effort to learner time ratio* are substituted.

[16] Senbetta (1991) used "Level of effort estimation for each hour of computer-based training" which he measured in "hours per hour". The paradigm definitions of *learner time* and *effort to learner time ratio* are substituted.

[17] Milette and Trevor-Deutsch (1995) used "Calculated CBT course length" measured in "hours", "Development ratio" measured in "number of hours required to develop one hour of CBT" and "Development time" measured in "hours". The paradigm definitions are substituted in the method.

[18] The range of effort to learner time ratios was calculated by entering the smallest and largest factors into the CEAC method. The number of steps was calculated by counting the number of alternative values for each factor. The total number of steps was calculated by multiplying together the number of alternatives for each factor.

[19] The step size is based on the assumption that most industry averages are rounded to the nearest 10 $\overline{Dh\ Lh^{-1}}$.

[20] The range of *effort to learner time ratios* was calculated by entering the 1 and 20 *Lh* into the regression method. The number of steps was calculated on the assumption that the learner time would be entered to the nearest five minutes or 240 steps.

[21] The range of *effort to learner time ratios* produced was calculated using the lowest and highest values of CD and DE which produce positive values of *effort*. The number of steps was calculated by summing the total number of combination of CD and DE which produced positive values of *effort*.

[22] The range of *effort to learner time ratios* was calculated by entering the smallest and largest factors into the quick and dirty method. The number of steps was calculated by summing the number of alternative values for each factor.

[23] The range of *effort to learner time ratios* was calculated by entering the smallest and largest factors into the *Q-factors* method. The number of steps was calculated using the assumption that *Q-factors* will be estimated to the nearest 0.1.

[24] The range of *effort to learner time ratios* was calculated by entering the smallest and largest factors into the US interactive courseware method. The number of steps was calculated by summing the number of alternative values for each factor.

[25] The range of *effort to learner time ratios* was calculated by entering the smallest and largest factors into the training cost calculation spreadsheet method. The number of steps was calculated by summing the number of alternative values for each factor.

[26] To compensate for the missing values at the top and bottom of the ranges, 1000 $Dh\ \overline{Lh^{-1}}$ and 50 $Dh\ \overline{Lh^{-1}}$ are assumed. The means of the band were used to calculate range of *effort to learner time ratios* for the CBT Analyst method. The number of steps is based on the fact that there are only four alternatives which directly affect the estimate.

[27] The range of *effort to learner time ratios* was calculated by entering the smallest and largest factors into the computer-based instruction length model method. The number of steps was calculated by counting the three alternatives, which directly affect the estimate.

[28] To compensate for the missing values at the top of the range, 600 $Dh\,\overline{Lh^{-1}}$ is assumed. The means of the band were used to calculate range of *effort to learner time ratios* for the Estimation grids method. The number of steps is based on the fact that there are only three alternatives which directly affect the estimate.

3. ESTABLISHING A MEASUREMENT PARADIGM

> "When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind." (Lord Kelvin 1889)

This now infamous quotation from Lord Kelvin comes from a period in the history of science when advances in the measurement of the physical world was resulting in real advances in pure and theoretical physics. It can be argued in Lord Kelvin's case that his insights into abstract thermodynamic concepts only became possible by advances in measurement and experimental techniques. While the glory may go to Lord Kevin, the credit must go to the numerous other scientists who carried out experiments, wrote papers and formed committees to define what was being measured and how it was measured. Over one hundred years later and in the area of sub-atomic particle theory the ability to construct models which predict invisible but measurable phenomena depends on physicists agreeing on what is being measured and how it is measured.

In software metrics there is very little agreement about what is being measured and how it is measured (Shepperd and Ince 1993). Using the results from a data collection experiment or a third party effort estimation model requires an act of faith from the computer scientist far greater than that required by a physicist in a similar situation. What are they using to measure size? Where do they start and stop measuring effort? What units are they using? These and numerous other questions have to be answered before any meaningful modelling can take place (Kitchenham et al. 1995).

What has all this to do with courseware effort estimation? In the previous Chapter the lack of consistent definitions severely hampered the development and analysis of courseware effort estimation methods. It is impossible to account for variations in the predictive accuracy of courseware effort estimation models if the experimental data contains results with different development life-cycles and includes or excludes different development activities. Agreeing what is being measured and how it is measured is an essential prerequisite for courseware effort model development. In this

Chapter, a measurement paradigm will be established under the following headings:

- Developers' opinion on courseware effort estimation
- The measurement paradigm
- Size measures
- Other measures
- Evaluating the measurement paradigm

First the results of a small scale survey will be presented to update our knowledge of courseware development and effort estimation.

3.1 DEVELOPERS' OPINION ON COURSEWARE EFFORT ESTIMATION

In Chapter 2 the results of previous surveys into courseware effort estimation were presented (Jay et al. 1987; Senbetta 1991). While they provided information and analysis of expert opinion into courseware effort estimation, their relative ages make them less valuable today. Authoring techniques and multimedia technology have advanced considerably in the five years since Senbetta (1991) published his survey and would be almost unrecognisable to the developers surveyed by Jay, Bernstein and Gunderson (1987) nine years ago. This Section presents the results of a small scale survey into multimedia courseware development and effort estimation.

3.1.1 Survey methodology

The literature review in Chapter 2 highlighted the inadequacy of the size and effort measures and methods currently used to describe multimedia courseware. This also showed that there was no general agreement about which tasks should be included nor about what methods might be used to prepare estimates. It was, therefore, not surprising that the self-reported and observed accuracy of effort estimates was very low (Jay et al. 1987; Senbetta 1991; Marshall et al. 1995c). To update these studies a questionnaire was used to gather data about current multimedia courseware development and effort estimation methods form courseware developers.

Survey

The four page questionnaire in Appendix 7.2 was designed using Logotron's Pinpoint for questionnaire design software. It consisted of twenty-two open and closed response questions relating to the respondents, their organisational background and effort estimation methods. Adequate space was provided for open responses and an optional area was provided for respondents to identify themselves. An ASCII file version of the questionnaire was then extracted from Pinpoint and posted by e-mail to Internet mailing lists during the summer of 1995. The mailing lists selected were related to development of multimedia courseware.

Respondents

A total of thirty-six questionnaires were returned by the end of the survey period. Initial validation and analysis was done in Pinpoint but the data were exported to Minitab for

more detailed analysis. Table 3.1 shows a summary of the courseware development experience of the respondents and their organisations.

Table 3.1 - Summary of respondents and their organisation's backgrounds

Respondent Number	Organisational Background	Number of Courseware Development Projects in Last 5 Years	Organisation's Multimedia Courseware Development Experience (Years)	Organisation's Other Courseware Development Experience (Years)
1	Non-commercial	2	5	10
2	Non-commercial	5	2	8
3	Non-commercial	5	2	5
4	Commercial	2	1	0
5	Non-commercial	10	2	1
6	Commercial	10	3	0
7	Commercial	25	4	4
8	Non-commercial	40	3	25
9	Commercial	6	5	45
10	Commercial	2	7	15
11	Commercial	30	4	100
12	Non-commercial	5	2	3
13	Non-commercial	10	7	7
14	Commercial	4	1	0
15	Non-commercial	12	3	0
16	Non-commercial	4	3	10
17	Non-commercial	15	2	8
18	Commercial	8	1	0
19	Non-commercial	3	5	20
20	Non-commercial	30	5	*
21	Non-commercial	3	1	15
22	Commercial	6	2	15
23	Commercial	12	*	6
24	Non-commercial	50	32	32
25	Commercial	4	1	0
26	Non-commercial	7	8	26
27	Non-commercial	20	10	30
28	Commercial	50	3	6
29	Non-commercial	5	5	*
30	Non-commercial	10	6	11
31	Non-commercial	*	12	18
32	Non-commercial	3	2	75
33	Non-commercial	4	3	5
34	Non-commercial	5	3	10
35	Commercial	*	1	1
36	Non-commercial	3	3	*

Analysis

Analysis of the results, of the survey is presented in two parts. The first part looks at the survey data in terms of four measures which characterise developers and their organisations. The distribution of these measures for commercial and non-commercial

organisations are summarised: in each case the values for commercial organisations are somewhat higher than for non-commercial although the differences do not reach statistical significance. These measures represent potential productivity adjustment factors which developers could use to evaluate their multimedia courseware development environment. The second part considers the results of the survey relating to multimedia courseware effort estimation.

3.1.2 Potential productivity adjustment factors

A number of existing multimedia courseware development effort estimation methods make use of various characteristics of the developers or the host organisation as productivity adjustment factors (Kearsley 1985; Gery 1987; Schooley 1988; Miles 1990; Golas 1993; Marshall et al. 1994d). Unfortunately, these productivity adjustment factors tend to be poorly defined or inconsistent and are normally based only on the author's opinion. Initial analysis of the data suggest that some characteristics of the respondents and their organisations could be described using the following productivity adjustment factors:

- Organisational goal
- Organisational experience
- Organisational sophistication
- Developer experience
- Management experience

These potential productivity adjustment factors were intended to provide a framework for discussion and development rather than the last word on the subject. The aim of defining these productivity adjustment factors was to gain some insight into the multimedia courseware development process and context which could be fed into effort estimation. Each of these five productivity adjustment factors are described below along with the significant results from the developer survey.

Organisational goal

Marshall et al (1995b) differentiated between commercial and non-commercial organisations in their framework for multimedia effort estimation. Their argument was that the organisational goal of a *commercial* producer was to make a profit and reduce

costs, while for a *non-commercial* organisation 'spending the budget' or 'personal satisfaction' were more likely to drive the development. Of the thirty-six respondents, fourteen (39%) described themselves as commercial while the remaining twenty-two (61%) described themselves as non-commercial. Non-commercial respondents were employed by educational, research, governmental or military organisations. Commercial respondents tended to work for developers or publishers of multimedia courseware or act as freelance consultants. The profit motive, be that personal or organisational, clearly divided the respondents into two groups. The *organisational goal* productivity adjustment factor had two alternative values; commercial applied when the aim was to make a profit and non-commercial applied in any other situation.

Organisational experience

The *organisational experience* productivity adjustment factor measures previous courseware development experience. On this basis the *organisational experience* productivity adjustment factor was constructed using scores related to the:

- Number of years of multimedia development experience
- Number of years of other courseware development experience
- Use of a rigorous estimating method
- Percentage contribution of multimedia to workload

Several existing effort estimation methods assumed that productivity was related to the number of years of multimedia development experience, and by extension, experience in developing any other types of courseware was also assumed to contribute to general productivity (Marshall et al. 1994d). Table 3.2 describes the minimum, mean and maximum years of development experience respondents reported their organisation had in developing multimedia courseware and other types of courseware.

Table 3.2 - Multimedia and other courseware development experience

Development Experience	Multimedia		Other	
	Commercial	Non-commercial	Commercial	Non-commercial
Minimum years	1	1	0	0
Mean years	5.15	4.59	16.00	15.42
Maximum years	32	12	100	75

As Table 3.2 shows, the maximum years for both multimedia and other courseware development are higher than expected. Investigation of the individual questionnaires

found that respondents included a pioneer of computer-based training who had been working in the area since the 1960's. Similarly, employees of a commercial publisher and a medical school reported their organisations had been developing other types of courseware for one hundred years and seventy-five years respectively. Taking these values into account the following scoring system was used.

- Organisations which have been developing multimedia courseware for more than 3 years were awarded five points
- Organisations which have been developing other types of courseware for more than ten years were awarded three points

An assumption of this productivity adjustment factor was that an organisation which had a rigorous method of estimating development effort was more experienced than those who did not. Since only 43% of commercial and 14% of non-commercial respondents claimed to have a rigorous method of estimating development *cost* or *effort* this appears to be a reasonable assumption. The following scoring system was used to allocate points.

- Organisations with a rigorous estimation method were awarded two points

The final assumption is that the greater the percentage of an organisation's work which was related to multimedia or computer-based courseware development, the more critical it would be to their success. The respondents were asked to select from one of four percentage ranges which best described the contribution of multimedia and computer-based training to overall organisational workload. The 'less than 1 %' range was not selected by any respondent and is excluded from further analysis. Table 3.3 describes the reported contribution of multimedia and computer-based courseware to overall organisational workload along with the score awarded for each range.

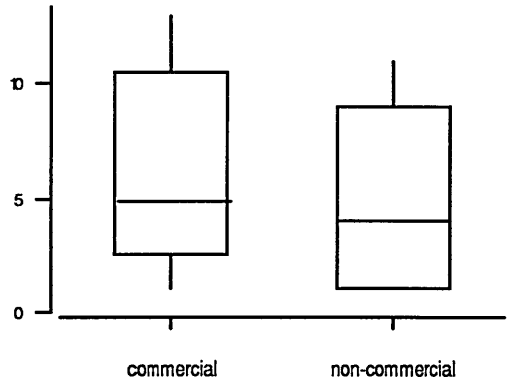
Table 3.3 - Multimedia as a percentage of organisation's workload

Organisational goal	**1% to 33%**	**34% to 66%**	**67% to 100%**
Commercial	3 (21%)	4 (29%)	7 (50%)
Non-commercial	17 (77%)	3 (14%)	2 (9%)
Contribution to *organisational experience* score	1	2	3

Using the sum of the four scores described above the *organisational experience*

productivity adjustment factor was calculated for each respondent. The distribution of *organisational experience* scores by *organisational goal* is shown in Figure 3.1.

Figure 3.1 - Distribution of *organisational experience* score by *organisational goal*



Organisational sophistication

It is reasonable to assume that an organisation which had made use of a range of different media, delivery techniques and courseware was more sophisticated than one which had only developed single medium, drill and practice courseware using a simple authoring system. The *organisational sophistication* productivity adjustment factor provided a measure which allowed comparison of the media and courseware delivery techniques and development methods used by different organisations. The productivity adjustment factor was constructed using scores related to the organisational use of:

- Courseware delivery methods
- Media in courseware development
- Courseware authoring tools

It can be argued that there was a vast difference in the *organisational sophistication* required to produce a high fidelity multimedia simulation than for a single medium certification test. Respondents were asked to identify all the courseware delivery methods their organisations had experience of developing. They were asked to differentiate between single medium which used just text or simple graphics and multimedia which also used audio, video, complex graphics or animation. A number

between one and five was assigned to each courseware delivery method the organisation had developed. Single medium drill and practice was assigned a score of one, single medium and multimedia high fidelity simulations were assigned a score of five and other types took values in between. Table 3.4 shows the score allocated for each courseware type and delivery method along with the percentage of respondents from commercial and non-commercial organisations who used each one.

Table 3.4 - Courseware type and delivery method score by *organisational goal*

Courseware type and delivery method	Score	Commercial (%)	Non-commercial (%)
Single medium - Drill and practice	1	36	64
Single medium - Certification tests	1	50	68
Single medium - Adaptive tests	3	21	27
Single medium - Tutorials	2	57	68
Single medium - Intelligent tutoring systems	3	36	27
Single medium - Exploratory environments	2	29	32
Single medium - Low fidelity simulations	3	50	45
Single medium - High fidelity simulations	5	29	18
Multimedia - Drill and practice	2	50	36
Multimedia - Certification tests	2	71	41
Multimedia - Adaptive tests	4	36	9
Multimedia - Tutorials	3	64	73
Multimedia - Intelligent tutoring systems	4	36	18
Multimedia - Exploratory environments	3	42	59
Multimedia - Low fidelity simulations	4	43	41
Multimedia - High fidelity simulations	5	50	18

It was also assumed that more sophisticated organisations would have made use of an extensive range of media types. The scoring system allocated one point to each media type used. Table 3.5 shows the percentage of organisations which had used each media type in previous developments.

Table 3.5 - Use of media type by organisational goal

Media type used in previous development	Commercial (%)	Non-commercial (%)
Text	93	95
Low resolution graphics	79	82
High resolution graphics	93	86
Photo-realistic graphics	79	68
Low fidelity animation	100	91
High fidelity animation	71	36
Simple representational simulations	71	71
Complex realistic simulations	50	45
Virtual reality	0	5
Digital audio	71	86
Digital video	36	41
Analogue (interactive) audio	21	41
Analogue (interactive) video	71	77

The questionnaire listed a range of alternative authoring tool types and respondents were asked to indicate the percentage contribution of each tool to overall courseware development. Table 3.6 describes the mean percentage contribution of authoring tool types to the respondent's organisational courseware development activities.

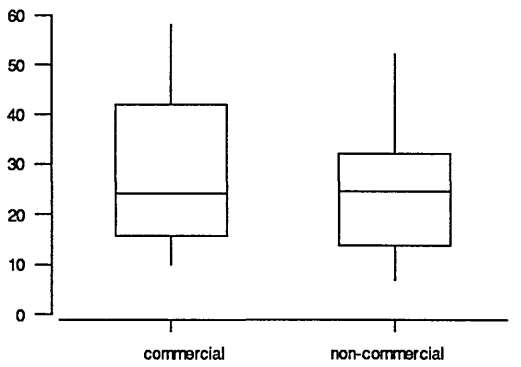
Table 3.6 - Mean percentage contribution of different authoring tool types to development effort

Authoring tool	Commercial	Non-commercial
Assembly language	0.4	0.1
General purpose programming language	15	17
Authoring language	35	48
Authoring system	40	35
Intelligent authoring system	9	0

Six respondents whose percentages did not total to 100% were omitted. As can be seen from Table 3.6, assembly language now only contributes a small percentage to the development effort as does the use of intelligent authoring systems. Surprisingly, there are no significant differences between commercial and non-commercial organisations in their reported use of authoring tools. However, only commercial developers claimed to use intelligent authoring systems and they also indicated that assembly language contributes four times as much to their development effort as it did to non-commercial developers (but the amount is still tiny). Authoring languages and authoring systems were the main contributors to development effort. The most frequently mentioned authoring languages used were Toolbook and HyperCard, whilst the most frequently mentioned authoring system was Authorware Professional. For a small, but significant proportion the courseware was developed using a general purpose programming language. The most frequently mentioned were C++, C and Visual BASIC.

Respondents who reported using intelligent authoring systems were awarded an extra point on the *organisational sophistication* score as were those who reported using assembly language. The sum of the values for the above factors gives the *organisational sophistication* productivity adjustment factor score. The distribution of the scores by organisational goal is shown in Figure 3.2.

Figure 3.2 - Distribution of the *organisational sophistication* score by *organisational goal*



Developer experience

The previous development experience of individual members or the team has been used as a productivity adjustment factor by a number of courseware effort estimation methods (Kearsley 1985; Gery 1987; Schooley 1988; Golas 1993; Marshall et al. 1994d). The underlying assumption was that the more years and the larger the number of projects a developer had been involved in, the more productive they would be. Similarly it was also assumed that the greater their experience the more accurate their estimates would be. It was also acknowledged that the experience of an individual developer may be distinct from that of an organisation and the developer experience productivity adjustment factor attempts to quantify this. This productivity adjustment factor was based on the following two aspects of *developer experience* measured over the period of last five years:

- Role in development teams
- Number of multimedia development projects worked on in last five years

The reason for selecting five years is primarily due to the rapid advance in technology and multimedia development methods.

An underlying assumption for developer experience is that a manager in a 'for profit' team is likely to be more experienced that the sole author in a 'not for profit' project. Respondents were asked to select one choice from six alternatives which best described their role in the courseware development projects over the last five years. Table 3.7 describes the results for commercial and non-commercial developers along with the score allocated for each role. For respondents who selected more than one alternative their highest score was chosen for the analysis.

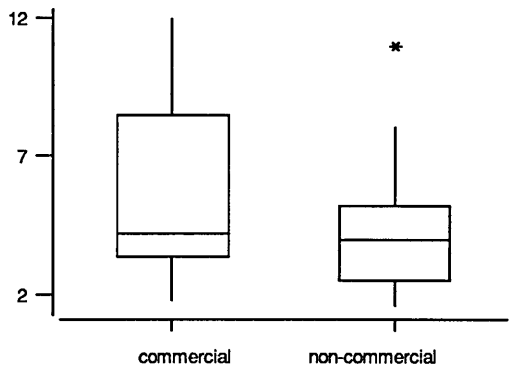
Table 3.7 - Percentage of respondents for each development team role

Role	Score	Percentage of commercial responses	Percentage of non-commercial responses
Sole author and developer on a 'not for profit' project	1	0	21
Sole author and developer on a 'for profit' project	1	0	5
Member of a 'not for profit' team	2	8	42
Member of a 'for profit' team	2	31	0
Manager of a 'not for profit' team	3	8	26
Manager of a 'for profit' team	3	54	5

The other assumption of this productivity adjustment factor was based on the number of projects the developer had been involved in over the last five years. The respondents produced a mean of 12.4 courseware developments over the last five years. Commercial developers tended to be more active than non-commercial developer producing a mean of 16.1 and 10.0 projects respectively. The contribution to *developer experience* was obtained by taking the number of multimedia projects undertaken in the last five years and dividing by five. This brought the contribution from this element to a similar range to that from the development team role.

Developer experience was calculated by adding together role score to the number of projects score. The distribution of *developer experience* scores is shown in Figure 3.3.

Figure 3.3 - Distribution of *developer experience scores* by *organisational goal*



Management experience

It is reasonable to assume that an experienced manager would be better at estimating than one with no experience. The *management experience* productivity adjustment factor was based on the following two elements measured over the period of the last five years.

- Number of multimedia courseware estimates prepared
- Typical team size

The more experience a manager has at managing the *effort* and *costs* associated with multimedia courseware development, the more productive the development team are likely to be. Respondents were asked to indicate the number of projects they estimated over the last five years. The mean number of project estimates produced by all respondents was 14.9. Commercial estimators produced a mean of 25.5 estimates compared to 7.4 for non-commercial estimates. Table 3.8 describes the contribution of estimation experience to the *management experience* score.

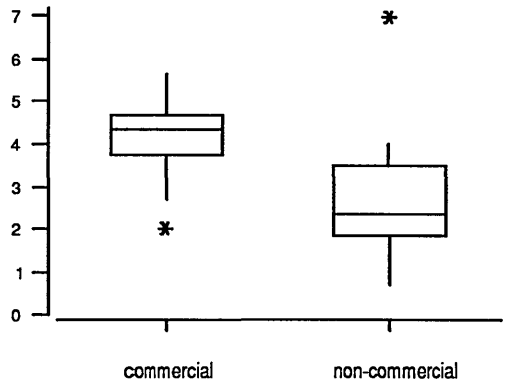
Table 3.8 - Contribution of estimation experience the *management experience* score

For multimedia projects in the last five years	Score
More than 10 estimates prepared	3
Between 6 and 10 estimates prepared	2
Between 1 and 5 estimates prepared	1
No estimates prepared	0

The assumption was that those who have been involved with larger teams were likely to have directly or indirectly received management experience. Respondents were asked to indicate the typical size of teams they had been involved with over the last five years. The mean team size for all respondents was 4.5 while commercial developers produced a mean of 5.3 in comparison to 4.1 for non-commercial teams. The typical size of the project team the respondent had worked with over the last five years is divided by three to bring the contribution from this element to the same range of values as that from the number of estimates.

The sum of the two elements gives the *management experience* productivity adjustment factor score. The distribution of the *management experience* scores by *organisational goal* are shown in Figure 3.4.

Figure 3.4 - Distribution of *management experience* scores by *organisational goal*



Correlation among productivity adjustment factors

One of the problems with the use of productivity adjustment factors in software effort estimation models is the high degree of inter-correlation. In effect, the productivity adjustment factors tended to end up measuring closely related or overlapping features of

the development process. It is therefore essential to establish the independence of the productivity adjustment factors suggested here. Table 3.9 shows the correlations among the four factors.

Table 3.9 - Correlations among the four productivity adjustment factors.

	Organisational experience	Organisational sophistication	Developer experience
Organisational sophistication	0.506		
Developer experience	0.143	0.602	
Management experience	0.091	0.197	0.259

The *organisational sophistication* score has quite large correlations with *organisational experience* and *developer experience*, but all the other correlations were small.

3.1.3 Productivity adjustment factors and effort estimation

Having defined the five productivity adjustment factors, they were then used to analyse the data collected about multimedia courseware development effort estimation from the respondents. The data collected on multimedia courseware effort estimation was analysed under the following headings.

- Use of a rigorous estimation method
- Description of 'cost'
- Base size measures used in estimation
- Tasks included in estimates
- Reported accuracy of estimates
- Range of development to learner time ratios
- Life-cycle phase distribution

Use of a rigorous estimation method

Table 3.10 shows the raw counts and percentages of commercial and non-commercial organisations which claimed to use a rigorous estimation method.

Table 3.10 - Use of a rigorous estimation method by commercial and non-commercial organisations

Organisational goal		Use of a rigorous estimation method		
		No	Yes	Total
Commercial	n (%)	8 (57.14)	6 (42.86)	14 (100)
Non-commercial	n (%)	19 (86.36)	3 (13.64)	22 (100)

Although the proportions of the two groups using a rigorous method were markedly

different, the difference failed to reach statistical significance because of the small numbers involved. There were no significant differences between those who did and those who did not use a rigorous method on any of the productivity adjustment scores *organisational experience, organisational sophistication,* and *developer experience.* However, those who did use a rigorous method had a significantly higher score on *management experience.* (The means are 4.11 and 2.65, $t=-3.20$, $p<0.01$)

Analysis of the written descriptions of the rigorous estimation methods described by respondents tended to focus on work breakdown, or on estimating the number of learner hours, or some other size metric. Using or failing to use a rigorous estimation method was the only factor that influenced reported estimation accuracy.

Reported accuracy of estimates

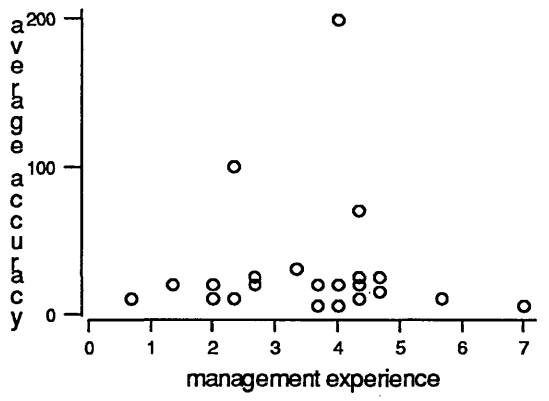
Respondents were asked to indicate on average how accurate their organisation's estimates were in comparison to actual effort. Table 3.11 describes the mean error in estimate as a percentage of actual effort and the worst over-estimates and under-estimates reported by respondents.

Table 3.11 - Range of error in estimated effort reported by respondents

Description	Commercial (Mean %)	Non-commercial (Mean %)
Worst percentage under-estimate of project *effort*	40	18
Estimates are normally within percentage of project *effort*	20	41
Worst percentage over-estimated of project *effort*	95	37

The reported accuracy of estimates did not depend on any of the productivity adjustment factors. Figure 3.5 below is typical of the results for the five productivity adjustment factors.

Figure 3.5 - Accuracy of estimates by *management experience*



The reported average accuracy did not depend on the number of tasks included in the estimates. There was no significant difference in mean reported average accuracy between commercial and non-commercial organisations. The mean reported average accuracy did not differ significantly for those who did and those who did not use each of the size measures listed in Table 3.13. However, it did for those who did and those who did not use a rigorous method. The means were within 11.43% of actual for those who did use a rigorous method and 37.8% for those who did not, ($t=2.33$, $p < 0.05$). It would, therefore, appear that the use of a rigorous method does have an impact on reported accuracy of development effort estimates.

Description of 'effort'

Within the literature there is some disagreement among authors as to how to measure the 'effort' associated with courseware development. Some authors use monetary value, others use effort while yet others favour *development time* or the number of developers involved. The questionnaire attempted to establish the dominant description of 'effort' used by respondents. Table 3.12 describes the alternative descriptions of 'effort' selected by respondents.

Table 3.12 - Percentage of respondents who ticked each description of effort

Description of multimedia courseware 'effort'	Commercial (%)	Non-commercial (%)
Monetary value	64	41
Effort in developer days	21	50
Number of developers involved	21	23
Other	21	14

It should be noted that some respondents (8, or 22%) indicated that their organisation used more than one description of effort. Two, or 6%, did not indicate any method of describing effort in their organisation (only one of these was non-commercial). None of the differences in proportions between commercial and non-commercial reach significance. There were no significant differences between those who did and those who did not use each of the measures on any of the productivity adjustment factor scores *organisational experience, organisational sophistication, developer experience, management experience.*

Base size measures used in estimation

All respondents, not just those who used a rigorous method, were asked to tick those base size measures they use in preparing estimates. Some selected more than one alternative. In addition, they were asked to indicate any other factors which they used to construct the estimates. A number of respondents did write in other measures but these were variations of the selection already described. Table 3.13 describes the range of base size measures used to make the initial estimate of the effort of a multimedia courseware development.

Table 3.13 - Percentages of respondents who ticked each size measure

Size measure used in estimation	Percentage of commercial	Percentage of non-commercial
Number of expected learner hours	50	32
Number of lessons	14	23
Number of objectives	14	27
Number of screens	50	32
Number of interactions in total	36	18
Number of interactions per hour	21	5
Number of frames	7	14
Number of media objects	43	41

There were no significant differences on mean *organisational sophistication* score between those who did and those who did not use each of the above measures. Those who used *number of objectives* had a higher mean *organisational experience* score than

those who did not and the difference only just failed to reach significance. (The means were 8.14 and 4.68, t=2.25, p=0.051). Those who used *number of expected learner hours* had a higher mean *management experience* score than those who did not, and this difference is significant. (The means were 3.95 and 2.60, t=2.72, p<0.05). Those who did not use *number of lessons* had a significantly higher mean *developer experience* score than those who did. (The means were 5.37 and 2.71, t=3.68, p<0.01).

Tasks included in estimates

In the literature there is disagreement among authors as to exactly what is included in multimedia courseware development estimates. Some authors specifically include tasks which others specifically exclude (Marshall et al. 1994b). All respondents were asked to indicate from a list of potential tasks exactly what they considered to be included in the estimates they gave. Table 3.14 describes the list of tasks they included in their estimates.

Table 3.14 - Percentage of respondents who included each task in the development cost estimate

Task	Commercial (%)	Non-commercial (%)
Preparation of estimate	29	27
Training needs analysis	50	46
Multimedia design	71	77
Courseware maintenance	36	50
Preparation of support materials	64	54
Programming costs	86	86
Learning about the subject matter	50	41
Client meetings	50	41
Video production	71	73
Audio production	71	68
Animation production	79	77
Graphics production	79	86
Media integration	71	54
Secretarial support	21	18
Revisions	57	54
Management	71	36
Copyright clearance	36	41
Other support	29	18
Technical reports	29	23

Commercial and non-commercial respondents both agreed on what tasks should be included in the production of an estimate. The only task for which there was a significant difference between commercial and non-commercial developers was *management* (χ^2= 4.21, p<0.05). This confirms the findings of Canale and Wills (1995)

who found that for non-commercial multimedia courseware producers in higher education there is a "...tendency to under-estimate the importance and cost of project management, which often becomes an unfunded contribution".

Those who included programming costs in their estimates had a significantly higher mean *organisational sophistication* score than those who did not. (The means are 27.8 and 17.0, $t=3.10$, $p<0.01$). There are no other differences in *sophistication* score. Those who did not include the cost of preparing the estimate in their costs had a significantly higher *organisational experience* score than those who did. (The means are 6.43 and 2.89, $t=3.55$, $p<0.01$). Respondents who included the cost of preparing support materials had a higher mean *organisational experience* score but the difference just failed to reach significance. (The means are 6.35 and 3.92, $t=1.96$, $p=0.06$).

Those who include the cost of preparing support materials in their costs had a significantly higher *management experience* score than those who did not. (The means are 3.70 and 2.40, $t=2.42$, $p<0.05$). Respondents who included the cost of media integration also had a higher mean *management experience* score but the difference just failed to reach significance. (The means are 3.60 and 2.48, $t=2.01$, $p=0.061$).

Range of effort to learner time ratios

The use of *effort to learner time ratios* is widely quoted in the literature as a method of estimating courseware development and the problems associated with its use have been explored in Chapter 1. However, to establish if this method was of any value, respondents were asked to indicate the maximum and minimum development effort for one hour of *learner time* for eight different types of courseware. The respondents were asked to indicate values for both single and multimedia courseware. Table 3.15 shows the mean minimum and maximum *effort to learner time ratios* for each type of courseware reported by respondents.

Table 3.15 - Mean minimum and maximum effort to learner time ratio

Courseware type	Single medium		Multimedia	
	Mean minimum ($\overline{Dh\ Lh^{-1}}$)	Mean maximum ($\overline{Dh\ Lh^{-1}}$)	Mean minimum ($\overline{Dh\ Lh^{-1}}$)	Mean maximum ($\overline{Dh\ Lh^{-1}}$)
Drill and practice	25	60	78	139
Certification test	33	102	61	137
Tutorial	86	212	125	292
Adaptive tutorial	112	316	126	338
Intelligent tutoring system	233	534	563	1357
Exploratory environments	178	446	254	493
Low fidelity simulations	120	279	239	480
High fidelity simulations	460	1230	504	1031

Many respondents failed to answer all or part of this question. For example, only six offered a maximum value for Single medium - Adaptive tutorial. Table 3.16 gives the full range of values reported from the smallest minimum to the largest maximum.

Table 3.16 - Range of reported effort to learner time ratio

Courseware type	Single medium		Multimedia	
	Smallest ($\overline{Dh\ Lh^{-1}}$)	Largest ($\overline{Dh\ Lh^{-1}}$)	Smallest ($\overline{Dh\ Lh^{-1}}$)	Largest ($\overline{Dh\ Lh^{-1}}$)
Drill and practice	1	300	3	600
Certification test	2	600	2	600
Tutorial	1	500	2	1000
Adaptive test	15	800	40	500
Intelligent tutoring systems	3	2000	150	5000
Exploratory environment	1	1000	10	1280
Low fidelity simulation	1	1000	2	2000
High fidelity simulation	4	5000	3	5000

The range of *effort to learner time ratio* reported in Table 3.16 for the same courseware type and media characteristic highlights the difficult in using it in effort estimation. The lack of standard definition as to what constitutes a tutorial, high fidelity simulation or adaptive test makes it difficult to draw conclusions from the extreme range of values supplied. In an earlier study, Marshall, Samson and Dugard (1994a) were able to explain some of the range of values produced by careful analysis of the techniques and terminology used by individual developers. Literally one developer's multimedia tutorial was another's single medium drill and practice courseware. Without clearly defined universal definition, it becomes difficult to draw meaningful conclusions.

Effort distribution over the life-cycle

The respondents were asked to indicate the distribution of effort spent during the nine phases of a multimedia courseware development. The intention here is to try and

establish the perceived distribution of effort over the life-cycle of the project. Table 3.17 describes the mean distribution of effort recorded by respondents.

Table 3.17 - Distribution of effort spent during multimedia courseware development

Phase	Mean commercial	Mean non-commercial
Initial analysis	10	8
Overall courseware design	13	12
Media design	8	9
Software programming	19	23
Media creation/sourcing	17	15
Software design	8	10
Media/software integration	12	8
Courseware evaluation	4	7
Testing/revision	9	9
Total	100	100

Four respondents whose total time did not add up to 100% were omitted as were four others who omitted all or part of the question. As Table 3.17 shows, there was no significant difference between the distributions reported by commercial and non-commercial organisations. Table 3.18 excludes the minimum, which is always zero, but shows the mean and maximum percentage of total development effort expended on each phase in the life-cycle.

Table 3.18 - Mean and maximum percentage of development effort by phase

Phase	Mean % of effort	Maximum % of effort
Initial analysis	9	20
Overall courseware design	13	40
Media design	8	30
Software programming	22	60
Media creation/sourcing	16	40
Software design	9	50
Media/software integration	10	50
Courseware evaluation	6	20
Testing/revision	9	30

The results presented in Table 3.17 and Table 3.18 are in line with those reported in the earlier surveys by Jay, Bernstein and Gunderson (1987) and (Senbetta 1991). The large range of values in any particular phase perhaps reflects the wide range of development techniques and delivery methods used in the construction of modern multimedia courseware. It also partially reflects the lack of consensus on the naming of phases and the allocation of tasks to these phases within the developer community.

3.1.4 Discussion

The survey confirms and updates the findings from earlier studies and, while the results were disappointing, these were not totally unexpected given the relatively small size of the survey sample. The five potential productivity adjustment factors do appear to characterise developers and their organisations in a less subjective manner than those used by exiting effort estimation methods (Kearsley 1985; Gery 1987; Schooley 1988; Miles 1990; Golas 1993; Marshall et al. 1994d). However, while these do assist understanding of the factors which affect productivity and hence development effort, they fail to reach significance in most of the areas investigated relating to estimation.

Even with the present unsatisfactory state of knowledge, many developers could improve their estimates by using a more rigorous method. This was the only factor which had an impact on the reported accuracy of estimates. However, there is no agreement about how 'effort' or size should be measured for multimedia courseware nor about which tasks should be included in an estimate. It can be argued that some of the problems associated with effort estimation could be reduced by the use of a consistent and widely agreed terminology.

The next Section in this Chapter establishes a measurement paradigm as the first step towards a common terminology. Without this effort estimation will remain inaccurate, because developers will not have a well defined method for describing what they are developing.

3.2 THE MEASUREMENT PARADIGM

The results of the survey confirmed that one of the major problems with existing effort estimation methods is the lack of agreement among developers and researchers on what is being measured. In this Section the measurement paradigm will be formally established using a top-down approach using the results of the survey and other studies into courseware development. The literature relating to the measurements defined by the paradigm will be explored before the following elements are described:

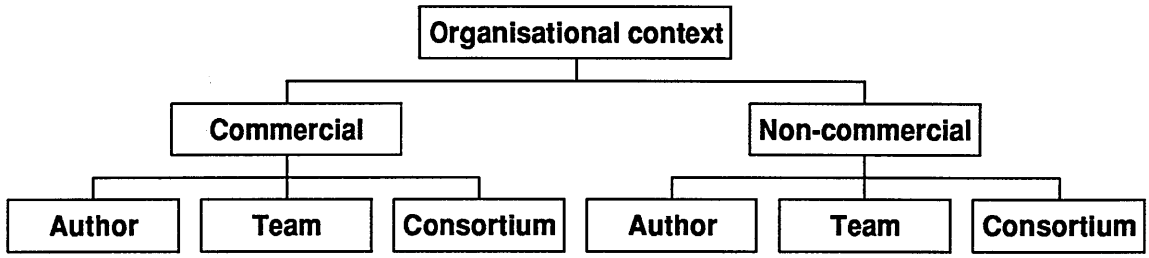
- Organisational context
- Courseware quality
- Development life-cycle
- Development activities

These measurement elements form the basis for the formal definition of size and other measures which will be discussed in later Sections.

3.2.1 Organisational context

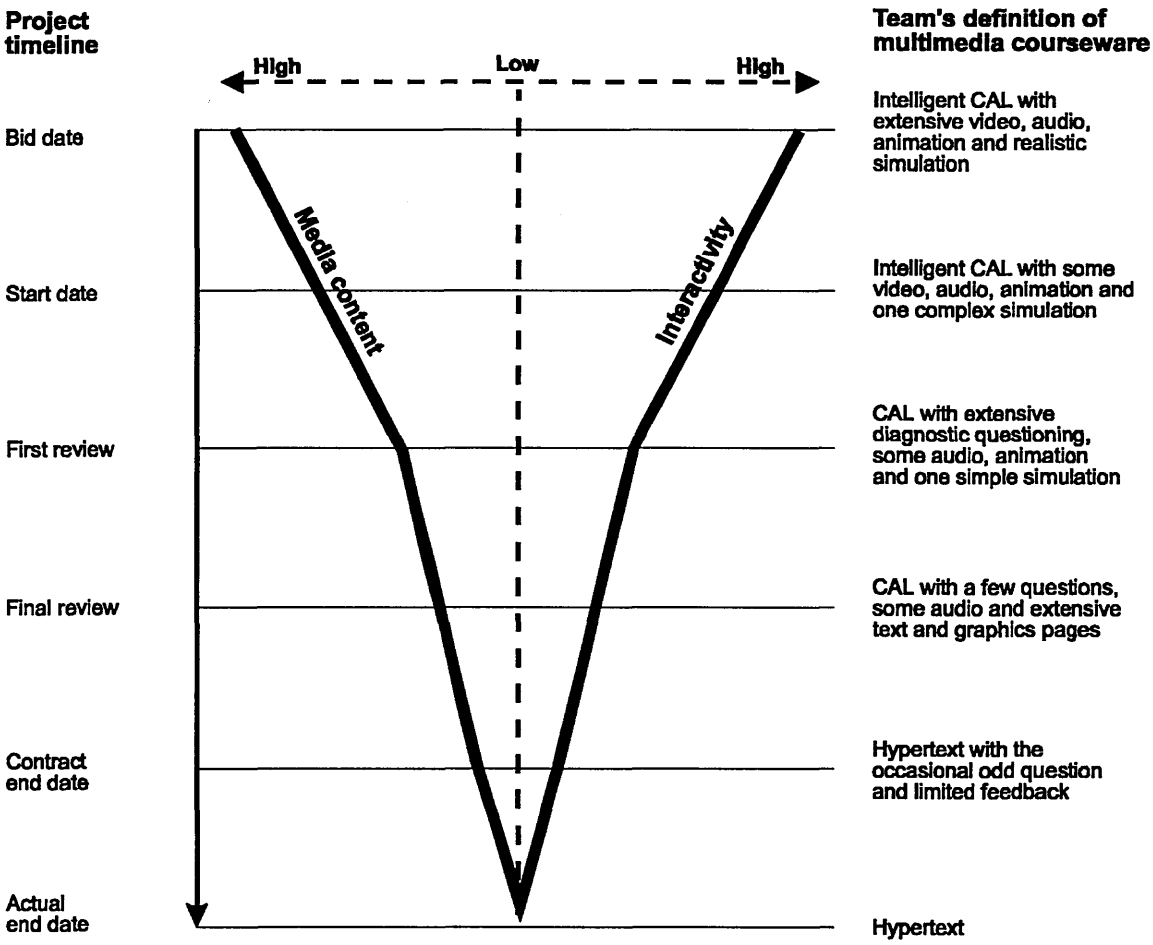
Organisational context describes the organisational environment in which multimedia courseware is developed. It is important to have a clearly defined organisational context to assist with the selection and calibration of courseware effort estimation models. Other than Golas (1993), none of the existing methods specify the source of the data which had originally been used to develop the method. They also did not specify the organisational context in which the method could be used. It has been argued that the range of *effort to learner time ratios* produced by the various methods may be due to differences among the organisational contexts in which the methods were developed and the sources of project data (Marshall et al. 1995a). Figure 3.6 describes two relevant organisational contexts.

Figure 3.6 - Organisational contexts of multimedia courseware development (Marshall et al. 1995a)



In a commercial environment, the courseware is either developed for an external client or for the general training and educational market with the aim of making a profit. On the other hand, non-commercial developers tend to create multimedia courseware for internal educational, training or research use without the primary profit motive. Even within the same organisational type the courseware may be developed by either individual authors, teams, or even consortia. In each of these situations different pressures exist on the developer. For example, in a non-commercial environment individual authors may develop multimedia courseware in their own time for their own use. They may choose to expend excessive amounts of effort to perfect the product or very little to produce something for the next class on the basis that, if it does not work, they are on hand to deal with the problem for the learner. Similarly, in non-commercial teams and consortium projects based on internal or external funding, a fixed budget is normally provided over a fixed period of time to deliver a certain amount of courseware. The desire to "spend the budget" may result in some extremely inefficient development methods being adopted (Hobbs and Price 1994). Alternatively, the quality, interactivity and media content of the final product can be reduced to meet the budget (Tennyson et al. 1995). Figure 3.7 shows this diagramatically

Figure 3.7 - Multimedia courseware development cone



While the multimedia cone is intended to be humorous, it is unfortunately based on personal experience and discussion with project managers of non-commercial developments. The multimedia cone effect is difficult to achieve in a commercial organisational context although not impossible. Fixed price contracts with well defined deliverables and penalty clauses reduce the scope in commercial work for removing interaction and media content in order to meet a deadline. With public release courseware, market demand for 'bigger, better and faster" multimedia removes the opportunity for a commercial organisation to reduce interactivity and media content to deliver on time; in this case overruns are more common (Hellman and James 1995).

Marshall et al. (1996) argue that the major difference between the two organisational contexts is that the desire to maximise profit is missing from non-commercial organisations. A commercial developer will try to minimise development and maintenance effort to maximise profit while producing an acceptable quality end

product. Non-commercial developers do not normally have this pressure and may be developing courseware for a number of other reasons. Given this difference, using a model developed in a non-commercial organisational context within a commercial organisational context is unlikely to produce reliable results. It is essential that the following two factors are used to clearly define the organisational context in which the model was developed:

- Organisational goal
- Development team structure

Clearly specifying these enables an appropriate model to be used and relevant data to be collected to validate or calibrate courseware effort estimation models

3.2.2 Courseware quality

Defining, measuring and evaluating courseware quality is a complex issue and, although very important, is beyond the scope of this research. A number of authors have tried to provide paradigms for measuring courseware quality and have discussed in more detail the associated problems (Bennet 1990; Reeves 1991; Reeves 1992; Ross and Morrison 1993; Barker and King 1994). However, the following two aspects of courseware quality need to be considered in terms of courseware effort estimation:

- Product quality
- Educational quality

Each of these have to be clearly defined for courseware effort estimation models and associated data collection.

Product quality

There are wide variations in the quality of multimedia courseware produced in both organisational contexts. A low quality product could be produced by a non-commercial author entering simple text and graphics into a hypertext system for classroom use. In this situation, less than perfect standards of presentation, robustness and bugs may be acceptable in the finished product. A commercial developer producing the same courseware would have to develop a higher quality product which was well presented, robust, and designed and tested to ensure it required little on-going maintenance.

In general, the existing courseware effort estimation methods do not clearly define the quality of their end product. Those that do normally do it by exception or include a sizing factor for a higher quality end-product (Kearsley 1985; Schooley 1988; Golas 1993). Of these, Schooley (1988) provides the clearest definition of courseware product quality. CEAC excludes development effort required for the following:

> "... a job/task analysis, MIL-STD 1379B documentation (e.g. student and instructor guides), MIL-STD 2167A documentation (e.g. test plans, test procedures). Finally, the data-base values do not reflect development [effort] devoted solely to the preparation of a courseware product for a commercial market." (Schooley 1988)

Unfortunately, there is little agreement across different methods as to the required courseware product quality. This highlights the difficulty of comparing existing courseware effort estimation methods. It is essential to define consistent quality standards to describe the multimedia courseware which is to be developed. While there are a number of well defined USA Military courseware standards, they tend to be over-specified for what is required; a much simpler definition of multimedia courseware product quality will suffice.

To this end, multimedia courseware is defined to be a product which involves computer control and presentation of two or more communication media for an educational or training purpose. The end product developed should include all of the following elements:

- Software which controls the learning environment
- Computer based presentation of one or more media
- Computer-based assessment of the learning process
- Learning support materials
- Installation materials and software
- Learner record system

These end products should be produced to a commercial quality standard and be in the

form of camera-ready or master copy quality. This defines the end product quality required for estimation and measurement of development effort. Using this end product quality definition, a multimedia courseware effort model would be primarily designed for commercial use. The model could be used by non-commercial developers provided appropriate allowances were made for the differences in end-product.

Educational quality

While it is a relatively easy task to specify the end product quality of the courseware, educational quality is more difficult to define. Since the objective of courseware is to provide a learning experience for the learners, it is important to ensure consistency in the educational quality of the end product. It is possible to expend large amounts of effort developing courseware which is only marginally effective in achieving its educational objective and positively discourages learners from using it (Soloman 1994). None of the existing courseware effort estimation methods define the educational quality of the courseware to be developed. The problem, as indicated earlier, is how to define and measure the educational quality required from the courseware?

Avner, Smith and Tenczar (1984) considered the effect of courseware authoring tools on productivity and quality using a composite measure based on "... student performance score, teacher ratings, peer reviews and classroom observation." Avner (1994) described the factors used in his earlier quality measure.

"The big problem was getting similar measures across different subject-matters and populations. The ultimate criterion is performance on future tasks after instruction, job, next related course. If that measure is available, you can measure it to construct a multivariate predictor from more easily accessed measures, like expert ratings of courseware, student performance on in-course exams, instructor ratings, observational measures like time-on-task, problem-solving efficiency. ... Other times you do the best you can to make the most use of the best measures in each specific situation. In one situation, course exams may be a really good test of skill needed for future practical use of the material being taught. In such a situation, exam scores would get heavy weighting. In another situation course exam scores may represent nothing more than a measure of the skill of students to do short-term learning of nonsense."

Accepting all the problems associated with defining a measure of courseware educational quality, it still must be done in order to provide part of the measurement paradigm. It must also be measurable within the time-scale of a pilot project and related to the educational effectiveness of the courseware. It is therefore proposed that educational quality should be defined by the following two related factors:

- Mastery score gain
- Completion rate

The mastery score gain is calculated on the result of a pre-test and post-test assessment of the courseware's learning objectives when taken by a sample of thirty learners for whom the courseware was designed. It is calculated using the following equation.

$$Mastery\ score\ gain = \frac{\sum_{i=1}^{n} Posttest\ score_i - Pretest\ score_i}{n}$$

Where $Pretest\ score_i$ is the percentage result of an appropriate test to assess the knowledge, skill or attitudes delivered by the courseware for an individual student. $Posttest\ score_i$ is the percentage result of an appropriate test given after the student has

completed the courseware. The value of n is the number of students who completed both the pre-test and post-test assessments. Positive values for the *Mastery score gain* indicate the courseware has achieved some of its objectives. Negative values indicate that the students know less when they finished using the courseware. Values of *Mastery score gain* range from -100% to 100% with 0% indicating that the courseware has had no effect on the mean mastery of the students completing.

The completion rate is the percentage of those starting the course who completed it. It is calculated using the following equation.

$$Completion\ rate = \frac{Number\ of\ posttest\ learners}{Number\ of\ pretest\ learners} \times 100\%$$

Where the *Number of pretest learners* is the number who completed the pre-test and enrolled on the courseware. The *Number of posttest learners* is the number of learners who completed the post-test after the courseware. The *Completion rate* ranges from 0% to 100%; the higher the percentage the more learners completed the course.

Acceptable values for mastery score gain are 50% or above and for completion rate greater than 90% (Orlansky and String 1979). All courseware development should be designed to achieve these values and effort measurements should include the testing, evaluation and amendments required to achieve these values.

3.2.3 Development life-cycle

Having defined the quality of the end product, it is important to agree the development life-cycle which produces the multimedia courseware. In their analysis of existing effort estimation methods Marshall, Samson and Dugard (1994c) found that, even when the methods defined a development life-cycle, they used different starting and finishing points. For example, Schooley (1988) specifically excluded training needs analysis which Bergman and Moore (1990) specifically included. Table 3.19 summarises the life cycle phases used by different courseware effort estimation methods and authors who explored other aspects of effort estimation.

Table 3.19 - Multimedia life-cycles used in effort estimation

Author	Phase										
	An	Id	Co	Cg	De	Mu	Md	Pr	Re	Ot	Ma
(Jay et al. 1987)		✓	✓	✓		✓	✓	✓	✓	✓	
(Senbetta 1991)	✓	✓			✓				✓		
(Gery 1987)	?	?	?	?	?	?	?	?	?	?	
(Schooley 1988)		✓	✓	✓		✓	✓	✓	✓		
(Kearsley 1985)	✓	✓	✓	✓		✓	✓	✓	✓		
(Miles and Griffith 1993)	✓	✓			✓			✓			
(Milette and Trevor-Deutsch 1995)	?	?	?	?	?	?	?	?	?	?	
(Tan and Nguyen 1993)		✓	✓	✓		✓	✓	✓		✓	
(Mikos et al. 1987)	✓	✓			✓				✓	✓	
(Golas 1993)	✓	✓	✓	✓		✓	✓	✓	✓		
(Philips Professional Media 1996)		✓	✓	✓		✓	✓	✓	✓		
(Bergman and Moore 1990)	✓	✓	✓	✓		✓	✓	✓	✓		

Where:

An = Analysis

Cg = Create graphics

Co = Content writing

De = Development

Id = Instructional design

Md = Multimedia development

Mu = Multimedia design

Ma = Maintenance

Ot = Other

Pr = Programming

Re = Review, testing and implementation

As Table 3.19 shows, there is no consistency in the inclusion or exclusion of phases in the studies analysed. The terms used to describe the phases are also inconsistent. For example, three authors use the very broad term "Development" to cover several of the phases other authors choose to describe individually. In two cases, despite careful analysis of the method, there appears to be no description of exactly what is being used as the basis for measurement. However, all authors agree that development effort estimation should exclude the "Maintenance" phase. As Miles and Griffith (1993) stated maintenance is "... regarded as a separate contract item. Most commonly it appears to be left out of the [effort] estimation...".

Measuring a moving baseline

It is very difficult to draw meaningful conclusions from the previous effort estimation studies and, indeed, most courseware development literature. By not defining the start

and end points for the development life-cycle or using consistent terminology to describe the phases, meaningful comparison of the results is impossible. The following three studies illustrate this point.

Jay, Bernstein and Gunderson (1987) asked respondents to indicate the effort required for eight phases of an courseware life-cycle for four different instructional methods. Table 3.20 shows the mean *effort to learner time ratio* and percentage of the total life-cycle reported by the fifty-two respondents for a nominal one hour of *learner time*.

Table 3.20 - Life-cycle *effort to learner time ratio* distribution (Jay et al. 1987)

Development activity	[*Effort to learner time ratios* ($\overline{Dh\ Lh^{-1}}$)]							
	Drill and practice		Tutorial		Simple simulation		Complex simulation	
	n	%	n	%	n	%	n	%
Instructional design	22.5	14	28.4	12	30.3	13	39.7	12
Writing content	33.9	21	36.8	16	40.4	17	60.3	18
Writing programming code	37.9	23	50.8	22	56.3	23	101.5	30
Creating graphics	14.0	9	18.0	8	30.5	13	42.1	12
Story-boarding and producing video	15.0	9	39.4	17	22.1	9	19.3	6
Reviewing and implementing in-house revision	17.7	10	27.3	12	31.8	13	44.0	13
Reviewing and implementing client requested revision	11.7	7	19.0	8	18.5	8	24.8	8
Other	11.3	7	9.3	4	12.1	5	11.3	3
Total	164.0	100	229.0	99	242.0	101	343.0	102

Jay, Bernstein and Gunderson (1987) found that there are no significant differences across the different types of courseware apart from the "Writing programming code" of complex simulations which accounts for 30% in comparison to about 23% for the other types of courseware. A potentially interesting conclusion, but how does it compare with Senbetta's (1991) results shown in Table 3.21 for estimates of percentage of effort across a four phase courseware development life-cycle?

Table 3.21 - Percentage effort for four phases of courseware development (Senbetta 1991)

Phase	Percentage of effort (%)		
	Low	Mean	High
Planning and needs analysis	5	15	30
Design	10	23	40
Development	30	48	95
Testing and debugging	0	14	30
Total	45	100	195

It is impossible to draw any meaningful comparisons without making very broad

assumptions about the equivalence of "Development" and "Writing programme code". Yet this is difficult because "Development" may or may not include "Creating graphics" and "Story-boarding and producing video".

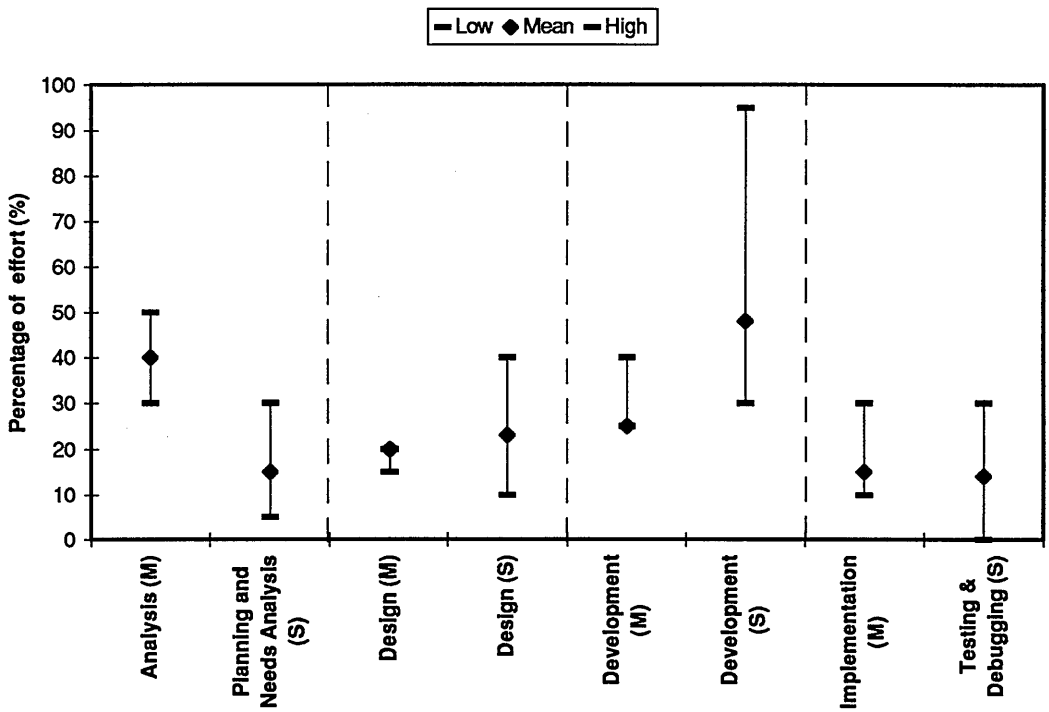
Even when authors agree on the number of phases that make up the life-cycle, comparison remains difficult. Miles and Griffith (1990; 1993) within their method used a five phase life-cycle. The phases consisted of analysis, design, development, implementation and maintenance. They stated that evaluation is distributed throughout all the phases and that the maintenance phase is excluded from the life-cycle. The resulting four phase life-cycle produces the reported percentage distribution of effort shown in Table 3.22.

Table 3.22 - Range of effort expended in each phase (Miles and Griffith 1993)

Phase	Percentage of effort (%)		
	Low	Recommended	High
Analysis	30	40	50
Design	15	20	20
Development	25	25	40
Implementation	10	15	30
Total	80	100	180

Both Table 3.21 and Table 3.22 have four phase life-cycles and even share two common phase descriptions with the other two phase descriptions indicating similar tasks and activities. Despite this, as Figure 3.8 shows, there are discrepancies in the distribution of percentage of *effort* reported between the two studies.

Figure 3.8 - Distribution of percentage *effort* by life-cycle phase description from two studies



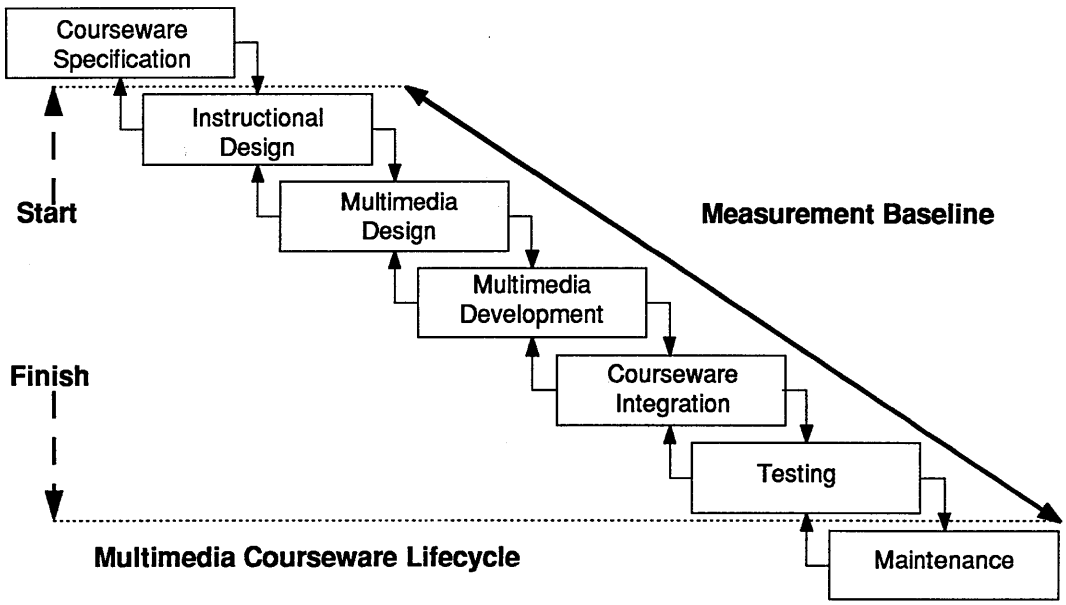
While a whole range of arguments could be put forward to explain the discrepancy, the ultimate conclusion must be that they are not measuring the same thing in similarly described phases. It is not even certain that they are starting or finishing the development at the same points. Miles and Griffith's (1990; 1993) "Planning and Needs Analysis" appears to include very early specification of the courseware and learners in comparison to Senbetta's (1991) "Analysis". The same applies at the end of the development where "Implementation" implies *effort* being expended beyond just "Testing and Debugging".

Proposed life-cycle measurement baseline

As the previous examples illustrate, there is no general consensus on phase descriptions or even the number of phases that makes up the courseware life-cycle. If that were not bad enough, there is no agreement on exactly where the life-cycle begins and ends. To create courseware *effort* estimation models it is essential that a common measurement baseline is established. This provides a firm foundation on which to measure and collect courseware data, to establish and validate *effort* estimation and other models.

Figure 3.9 shows a waterfall life-cycle model for multimedia courseware development. This is based on a number of existing Instructional Systems Design and Courseware Engineering models (Tennyson 1995). Although highly stylised and perhaps not truly reflecting the concurrent and iterative nature of multimedia courseware development, it serves as a basis for measurement of courseware developments.

Figure 3.9 - The waterfall model of multimedia courseware development



Outline phase definition

In the courseware specification phase the detailed objectives and outline content of the course are defined along with target audience and other information about the course to be delivered. Following this phase the overall instructional design is developed based on discussions with subject matter experts, courseware designers and the client. Once this phase has been agreed with the client, detailed design is undertaken for the media to be used in the final course. Additional experts such as media specialists, graphic artists, design editors and programmers may be added to the team at this point.

In the multimedia development phase the graphics, audio, video, sound and programme structure is developed. The courseware integration phase brings the various multimedia elements together on one platform. The testing phase involves pilot testing the educational quality and reliability of the courseware with learners. Problems identified

during the testing phase are fed back to improve the courseware quality. Once testing is complete, the product is ready to be published. The final phase is the maintenance phase in which the courseware is amended if serious problems are detected or if the client requires changes to be made to update the content, method or presentation.

The measurement baseline is defined to start at the beginning of the instructional design phase and to finish after the testing phase. The reason for excluding the courseware specification phase from the agreed development life-cycle is that, in a commercial development, the duration of this phase can vary considerably between projects. The client and developer are typically involved in negotiations as to what can be developed. This can be protracted or instantaneous depending on the differences between the client's expectations and the developer's cost base. By the end of the courseware specification phase, the developer and client will have agreed the major objectives, content, delivery platform, method and media for the project. Similarly, the duration of the maintenance phase is difficult to define accurately. Estimation of maintenance *effort* for multimedia courseware is a separate issue which may be worthy of further research.

3.2.4 Development activities

Development activities describe individual or on-going activities which form part of the courseware development. An activity may be specific to an individual phase, or appear in several phases of the life-cycle, or be required as part of the project support or management infrastructure. As with the phases in the life-cycle, there is no agreement in the literature as to which development activities are included in *effort* estimates. The comparison of models or experiments when different development activities are randomly included or excluded is difficult. In this Section the activities which will normally be included in the measurement baseline will be identified.

What the experts say

Using the predefined list of activities, respondents to Jay, Bernstein and Gunderson's (1987) survey were asked to identify those activities they usually included in courseware *effort* estimates. Table 3.23 shows a frequency ordered list of activities.

Table 3.23 - Frequency of activity inclusion in estimates (Jay et al. 1987)

Activity	Frequency included (%)
Writing lessons	86
Front end analysis	82
Revisions	79
Developing graphics	78
Programming lessons	77
Learning content	68
Programming routines	62
Management time	60
Formative evaluation	54
Meetings	52
Secretarial and other support	45
Summative evaluation	42
Video production	35
Computer operation	34
Technical reports	23
Computer down-time	11

Jay, Bernstein and Gunderson (1987) found that experienced[1] developers included "management time" more frequently (66%) than inexperienced developers (43%) (χ^2=5.0, $p<0.05$). Significantly more experienced developers included "developing graphics" more frequently (83%) than did inexperienced (65%) (χ^2= 4.4, $p<0.05$). Private organisations were much more likely to include "management time" as part of their estimate (68%) than were academic organisations (32%) (χ^2=12.3, $p<0.01$). The inclusion of "learning the content" in the estimate was more common among private organisations (72%) than academic (50%) (χ^2=4.8, $p<0.05$). There was no significant difference in the inclusion or exclusion of an activity by organisations which claimed to make accurate[2] or inaccurate estimates.

In his Delphic study Senbetta (1991) found discrepancies in the range of activities included in estimates by experienced courseware developers. Table 3.24 describes the activities respondents specifically included or excluded in courseware *effort* estimates.

Table 3.24 - Activities included or excluded in courseware *effort* estimates (Senbetta 1991)

Activity	Included		Excluded	
	n	%	n	%
Prototype/template building time	20	80.00%	5	20.00%
Production time and cost	18	72.00%	7	28.00%
Client content review/approval time	16	64.00%	9	36.00%
Evaluation (pilot test) participant's time	16	64.00%	9	36.00%
Subject matter expert's (SME) time	16	61.54%	10	38.46%
Administration/clerical support time	14	58.33%	10	41.67%
Hardware/software purchase or rental	13	54.17%	11	45.83%
Cost of external consulting assistance	12	48.00%	13	52.00%

Neither of these studies produced clearly defined lists of activities of what should or should not be included in all projects to be measured for estimation purposes.

Proposed development activity measurement baseline

Given the potential for confusion related to the activities which are included or excluded for measurement purposes, it is essential to have a simple definition of activities which count and those which do not. The *effort* from the development activities is included if:

- It occurs from the start of the instructional design phase but before the end of the testing phase
- It is carried out by a member of the internal project development personnel and management or external consultant or sub-contractor
- It is directly related to the development of the courseware, tools or methods required to deliver the current project

A development activity is specifically excluded if:

- It is carried out by the client organisation's employees or target learners
- It is carried out by management, administrative, secretarial, sales, marketing or other support staff not directly working on the project

The aim of these definitions is to localise the measurement within the scope of the current courseware development. For example, a programmer who uses a predefined template would only count the *effort* spent finding, learning, amending and using the template related to the current project. The *effort* expended creating the original template is not counted unless it is specifically developed for the current project. In a

similar way, organisational support and general management would not be counted. This is a general overhead which would be accounted for once the *effort* was converted into *cost* by use of an overhead rate.

One of the most difficult areas is that of activities which involve the client employees and target learners. In general, any activities which only involve these two groups are excluded. However, one exception is if the client is providing the Subject Matter Expert (SME). In this case, the SME is effectively acting as a sub-contractor and should be treated as part of the development team as far as those activities are concerned which relate to the project.

This concludes the formal definition of elements which form the measurement paradigm. They provide a consistent base on which to collect data and build courseware *effort* estimation models. They will now be used to define a consistent set of direct and indirect measures for courseware *effort* estimation research.

3.3 SIZE MEASURES

The identification of a size measure is important in accurate construction of estimates of development *effort* (Marshall et al. 1995b). In software *effort* estimation research the dominant size measure is source lines of code (SLOC). Despite numerous concerns being expressed about its relevance in early life-cycle estimation, how it is measured and its accuracy, it has the advantage of being a relatively simple measure which can be automated. Fortunately, or unfortunately depending on your perspective, its use as the basic size measure in courseware *effort* estimation is unknown. This perhaps reflects the general desire by early authoring system developers to remove the need to learn programming from the author and as a result the code is hidden from direct view. In the case of courseware *effort* estimation models the dominant size measure is *learner time*.

Jay, Bernstein and Gunderson (1987) in their survey asked one hundred and forty six respondents to identify what they used to estimate courseware size; 27% used *learner time*, 9% used "number of lessons", 7% used "number of screens required", and 15% used some other single measure such as "complexity of the content" or "number of interactions". The remaining 42% reported using some composite means which used more than one base size measure. Significantly, only 18% of inexperienced developers reported using a composite base size measure while 82% of the experienced developers used a composite base size measure (χ^2=3.69, $p<0.05$). There was also a significant difference in the use of composite base measures by private developers who used them more frequently than did academic (χ^2= 4.1, $p<0.05$).

In the same study respondents were asked their opinion on size measures for measuring courseware developments. Only 59% of the respondents favoured the adoption of an industry-wide standard method of measuring the size of courseware. Of the seventy-eight who replied 36% favoured the use of a mathematical formula, 10% the development *effort*, 6% the number of interactions, 5% the number of screens, 3% the number of lessons, 1% the *learner time*, 21% other unique units and 18% reported that they did not know a method of measuring courseware size.

The size measures will be discussed under the following headings:

- Learner time
- Screens
- Interactions
- Lessons
- Performance objectives
- Media objects
- Composite courseware measures
- Software-based

3.3.1 Learner time

Mikos et al. (1987) provided respondents with *learner time* as the size measure in their *effort* estimation exercises. However, there are no formal definitions of what was being measured by *learner time* (Jay et al. 1987). Some developers base their estimates on how long they or their client feel an average student will take to complete the courseware. Other developers define it once the product has been developed by testing it on an average student to produce a *learner time*, other developers suggest using the average *learner time* of a sample group of learners, whilst yet other developers described it in terms of the equivalent traditional instruction the student would receive in an hour. However, these vague definitions are almost meaningless which led Fairweather and O'Neal (1984) to characterise *learner time* as "... the most slippery measure known to man." However, Senbetta (1991) in his Delphic study found that twenty (76.9%) of the twenty-six respondents use *learner time*[3] to quantify the size of the courseware to be developed.

Experimental studies

One of the earliest studies of *learner time*[4] was carried out by Gailey (1973a; 1973b). The study investigated the construction and use of four courseware modules covering "Household Electricity" topics. IBM Coursewriter III was used to develop the modules for delivery on an IBM mainframe timesharing system. Gailey (1973a; 1973b) recorded the *learner time* spent using the modules by a group of students. Only twenty-four (45%) of the fifty-three students who started the evaluation completed all the modules.

Table 3.25 shows the range and mean *learner time* for individual modules and total for all four modules.

Table 3.25 - *Learner time* for individual modules and total course (Gailey 1973b)

Module	Learner time (*Lh*)			
	Low	High	Mean (n=53)	Mean (n=24)
A Electricity terms	0.70	3.92	1.50	2.03
B Safe use of electricity	0.50	0.97	0.63	0.62
C Household circuit game	0.30	2.63	0.77	0.80
D Cost of using electricity	0.32	0.63	0.40	0.39
Total	1.82	8.15	3.30	3.84

Courseware is intended to deliver individualised instruction in which the learners progress at their own pace. Which begs the question how is the *learner time* measured? The mean could be used but which one? As Table 3.25 shows, comparing the mean for all fifty three students who started the course to the twenty four who finished produced a 0.54 *Lh* (16.4%) difference.

More recently Orey et al. (1994) studied the development and use of intelligent computer aided learning (ICAL) courseware. Table 3.26 describes the *learner time* spent by individual participants using the courseware. Although it was not stated in the original study, it appears that five (41.7%) of the original twelve participants did not finished the courseware. If mean *learner time* is calculated using all twelve participants the result is 0.34 *Lh* in comparison to 0.59 *Lh* for the seven who actually completed the courseware. The differences between the two mean values of *learner time* is 0.25 *Lh* (42.37%).

Table 3.26 - *Learner time* spent by individual learners (Orey et al. 1994)

Student Number	Computer Experience (Years)	*Learner time (Lh)*
1	1.5	0.56
2	5	0.74
3	12	0.52
4	3	0.54
8	0	0.69
11	3	0.38
12	2	0.70
Mean	3.79	0.59

Table 3.26 also highlights a problem with the composition of the sample used to collect the *learner time* data. How does the sample compare with the population of the intended students? Do they have the range of computer and other experience described in Table

3.26? If they do not, then how reliable a predictor of actual *learner time* is the result from the pilot sample?

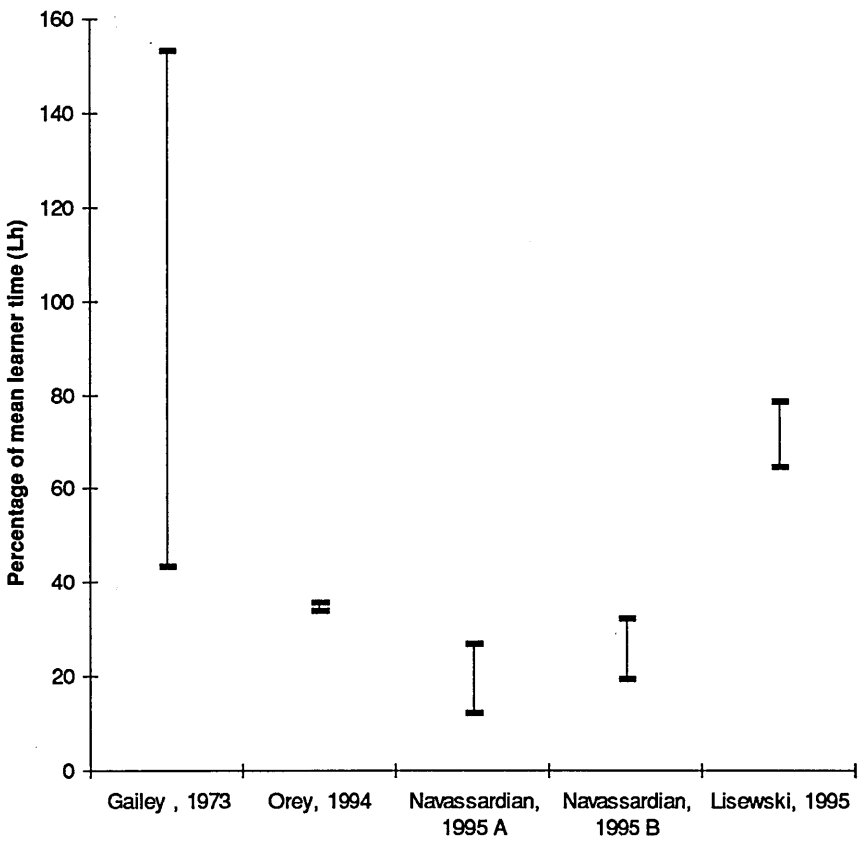
Using mean *learner time* also gives no indication of the range of individual *learner times* produced by the courseware. As Table 3.27 shows, using mean *learner time* gives little indication of the highest and lowest *learner time* produced by the courseware.

Table 3.27 - Percentage of mean in highest and lowest values of *learner time* from mean

Study	Range of *learner time*		
	Mean (*Lh*)	Low (% of Mean)	High (% of Mean)
(Gailey 1973b))	3.22	43.48	153.11
(Orey et al. 1994)	0.59	35.59	33.90
(Navassardian et al. 1995) 1	0.41	12.2	26.8
(Navassardian et al. 1995) 2	0.31	19.4	32.3
(Lisewski and Settle 1995)	14.9	78.5	64.4
Mean	3.89	37.83	62.1

The five studies found in the literature may not be representative but they do give an indication of the range of *learner times* reported in evaluation experiments. Care would have to be taken to ensure that the sample of students used to evaluate the *learner time* was representative of the population of students who would normally be expected to use the courseware. Figure 3.10 shows the ranges of *learner times* reported by the five experimental studies.

Figure 3.10 - Range of *learner time* reported by experimental studies



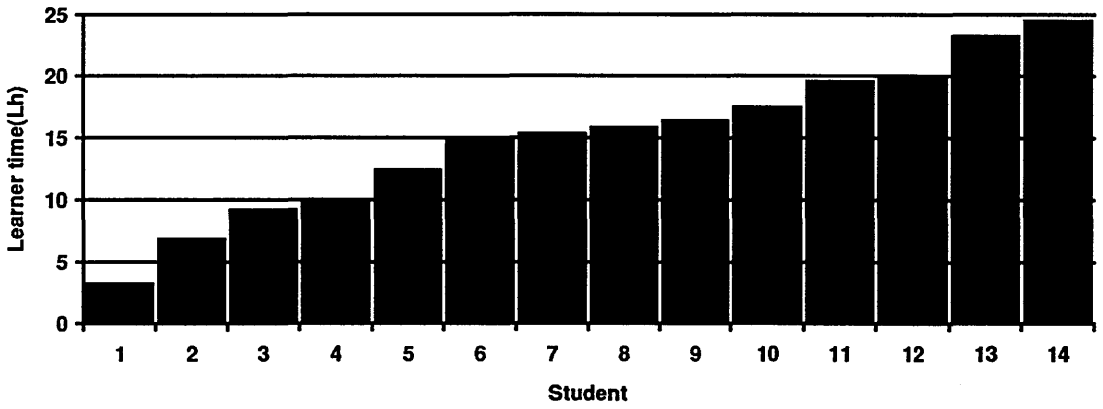
Estimating learner time

Senbetta's (1991) respondents identified three 'rules of thumb' they used to estimate *learner time*. These involved estimating the number of objectives, screens and interactions an average student would complete in one hour of *learner time*. Using simple definitions of each alternative base size measure, 76% of the respondents equated 3 to 8 objectives and 52% of the respondents equated 76 to 125 screens to an hour of *learner time*. Unfortunately, there was no general consensus on the number of interactions which constitute one hour of *learner time*.

Another commonly found technique is to use the traditional instruction equivalent lecturer time to estimate the *learner time*. However, as Navassardian, Marinov and Pavlova (1995) found in their comparative investigation of ICAL courseware and lecture-based instruction this is not equivalent. The students in the control group required two 1.5 hour lectures to achieve equivalent results to 0.72 mean *Lh* for the

courseware group. The two courseware modules produced a compression in the mean *learner time* to 27.33% and 20.67% of the traditional equivalent. Lisewski and Settle (1995) presented similar results when discussing an experiment to replace 20 one hour lectures with CD-ROM-based multimedia courseware developed for the Apple Maciintosh. The profile of student usage is shown in Figure 3.11.

Figure 3.11 - Total individual learner time by student number (Lisewski and Settle 1995)



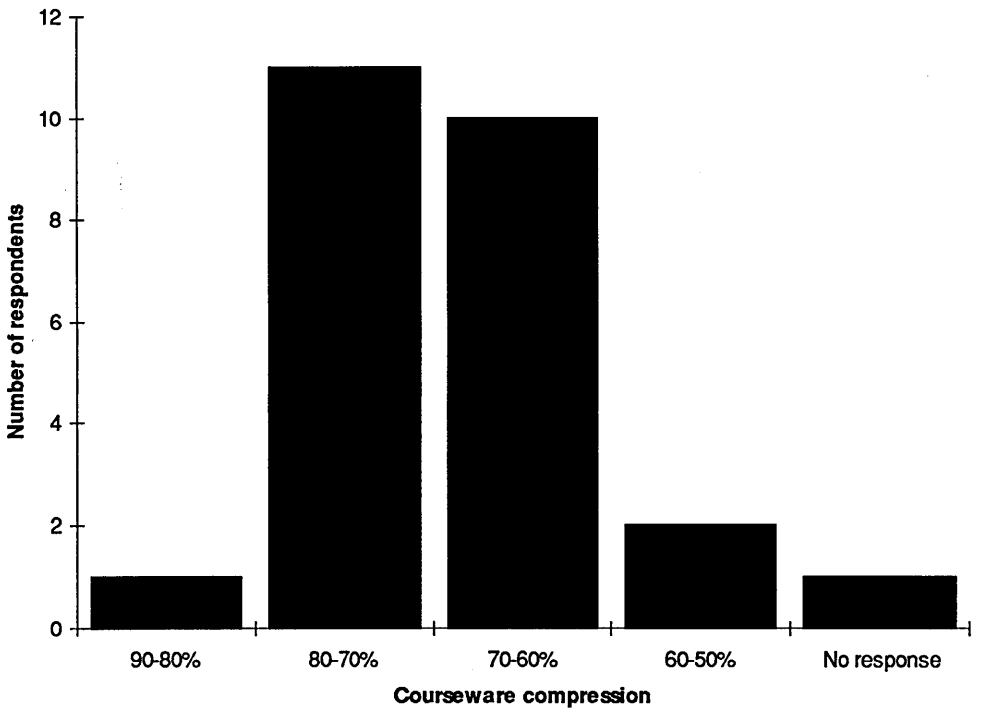
The mean *learner time*[5] for the 14 students was 14.9 *Lh* which is a 74.5% *learner time compression.* However, the quickest student took just 3.2 *Lh* and the slowest 24.5 *Lh.* These represented *learner time compression* of 16.0% and 122.5% respectively. While it is normal to assume a reduction in *learner time* in comparison to the traditional equivalent, the opposite is also possible. Orlansky and String (1979) found that the use of courseware increased *learner time* in three (7.50%) studies but had no effect in one (2.50%) other. Whatever the effect the following equation can be used to estimate the *learner time.*

$$\textit{Learner time} = \textit{Learner time compression} \times \textit{Traditional equivalent hours}$$

Several of the existing courseware *effort* estimation studies make direct or indirect use of *learner time compression.* For example, Milette and Trevor-Deutsch (1995) recommend the use of *a learner time compression* value of 50% in their method but indicate that they have been able to achieve an effective 25% *learner time compression* with some courseware. Schooley (1988) makes use of a 75% *learner time compression* in the example provided which appears to be loosely based on Orlansky and String's

(1979) results. Senbetta (1991) in his Delphic study asked respondents to estimate the reduction in *learner time* for a traditional course converted to courseware[6]. In this study 84% of the twenty five respondents suggested using a value for *learner time compression* between 80% and 60%. Senbetta's (1991) choice of overlapping ranges makes analysis of the results slightly more difficult than if disjoint ranges had been used. However, analysis of the results produced a mean *learner time compression* of 78.75%. Figure 3.12 shows the range of *learner time compression* percentages reported by respondents.

Figure 3.12 - Estimated *learner time* compression (Senbetta 1991)



As Table 3.28 shows a figure of between 50% and 80% is normally reported for mean *learner time compression* in the literature.

Table 3.28 - *Learner time compression*

Study	Method	Learner time compression (%)		
		Mean	Best	Worst
(Milette and Trevor-Deutsch 1995)	Author opinion	50	25	-
(Schooley 1988)	Author opinion	75	-	-
(Vogt 1995)	Author opinion	50	-	-
(Zimmerman 1988)	Author opinion	50	-	-
(Senbetta 1991)	Delphic study	78.75	55	85
(Lisewski and Settle 1995)	Experimental	74.5	16	122.5
(Roblyer et al. 1988)	Experimental	70	-	-
(Orlansky and String 1979)	Meta-analysis	71	20	131
(Kulik and Kulik 1986)	Meta-analysis	60	-	-
(Hirschbuhl 1989)	Survey	60	50	70

While the mean values in Table 3.28 may reflect actual developer experience, the results for surveys and author experience may reflect the dominance on the collective psyche of the widely quoted Orlansky and String's (1979) figures. Based on the available evidence Orlansky and String's (1979) recommendation of a 70% *learner time compression* for estimation purposes still seems reasonable. However, it should be remembered that this is a mean value and that, while some students will progress very quickly through the courseware, others may take longer than with traditional instruction.

Weaknesses of learner time

As well as the previously highlighted problems, there are a number of well known weaknesses in the use of *learner time* as the base size measure. As currently used, *learner time* does not measure the quality of the courseware and may even discourage good design (Jay et al. 1987). Good courseware includes extensive branching and detailed feedback to support weak students. Most of this may never be seen by the average or bright student and could be removed to simplify the development process without affecting the mean *learner time*. *Learner time* can also be artificially increased by padding the courseware with relatively cheap to produce page-turning screens which take a predefined amount of time to display.

Strengths of learner time

Learner time does have a number of strengths which explain its longevity as the preferred courseware size measure. Its main strength is that it is simple to explain to potential clients who typically wish to replace a certain number of hours of

conventional classroom-based instruction. It is also in theory relatively easy to measure although there is little evidence that it is actually measured. Most developers and clients use an expected number of learner hours and leave it at that. Nine out of ten of the existing courseware *effort* estimation methods use *learner time* as their base size measure. In addition, Jay, Bernstein and Gunderson (1987) investigated the use of alternative base measures, such as screen interactions and frames, for courseware development but found similar problems with them. So despite its obvious limitations, it would appear that *learner time* is firmly established as the base size measure for courseware.

Definition

While *learner time* has problems, it is widely used and accepted by the industry. To reduce the problems, it is essential to define a standard method for measuring *learner time*. Prior to the development it should be based on the time an average learner will be expected by the developer to complete the course. Once the courseware has been developed, *learner time* should be measured using the mean time taken for a representative sample of thirty learners for whom the material was developed to complete the course. The mean should be calculated using the learners who completed the pilot study post-test courseware assessment described previously.

3.3.2 Screens

Early courseware tended to be dominated by instruction which was based on providing screens, pages or frames of information based around an electronic book metaphor. The student would be presented with a sequence of screens containing text, graphics, simulations or questions. The exact ordering of the sequence depended on the student's choices or answers to questions designed by the courseware author. There is some support among the developer community for screen counts to be used as a size measure on the basis that it is *learner time*-independent (Jay et al. 1987). It sounds simple to just count the total number of screens in the courseware but, with the introduction of a windowing environment, hyper links, sound, animation and video elements, defining and counting screens becomes more difficult. The problems in accounting for differences in the following led Jay, Bernstein and Gunderson (1987) to reject screen counts as a size measure:

- Screen design and layout
- Media content
- Complexity of interaction

Other authors have explored screen counts. Senbetta (1991) found that 52% of experts in his Delphi study used between 76 and 125 screens to define one hour of *learner time*. To overcome some of the problems with this measure, the participants were asked to assume they were developed using good design principles. The difficulty with the definition of *learner time* reduced the value of this result and may account for the extreme range of values suggested by the respondents shown in Table 3.29.

Table 3.29 - Number of screens per hour of *learner time* (Senbetta 1991)

Number of screens per hour of [*learner time*]	Number of respondents	%
25-50 screens	1	4
51-75 sceeens	2	8
76-100 screens	7	28
101-125 screens	6	24
126-150 screens	1	4
151-175 screens	2	8
176-200 screens	2	8
200+ screens	3	12
No response	1	4

Cates (1994) used 16 novice designers and developers to carry out a small scale experimental study into courseware development. The courseware was designed to run on a simple IBM personal computer with Colour Graphics Adapter (CGA) display and twin 360k floppy disk drives using the SAM IV authoring system. As part of the study the number of post-development measures, including number of screens, was recorded. A screen was defined to consist "... of all the information that a learner would see on a single screen without clearing the screen" (Cates 1994). In addition, components which overlay on the screen were counted. A component was defined to be "... blocks of text in various typefaces, animation sequences, graphic images, interactive items that call for a response from the learner, or feedback messages that are specific to a learner response" (Cates 1994). Table 3.30 and Table 3.31 show the number of screens and components for each lesson along with *effort*[7] and *learner time* per screen and per component respectively.

Table 3.30 - *Effort* and *learner time* per screen (Cates 1994)

Lesson name	Screens	[*Effort*]		[*Learner time*]	
		[Dh]	Per screen	[Lh]	Per screen
Effectiveness + Introduction	52	177	3.40	0.42	0.0081
Situational leadership	85	531	6.25	0.63	0.0074
Distributed management in education	66	323	4.89	0.53	0.0080
Hoy-Miskal-bridges	93	331	3.56	0.65	0.0070
Lewin forces	63	237	3.76	0.48	0.0076
Other	15	*	*	*	*
Unit mean	75	320	4.28	0.54	0.0072

Table 3.31 - *Effort* and *learner time* per component (Cates 1994)

Lesson name	Components	[*Effort*]		[*Learner time*]	
		[Dh]	Per comp.	[Lh]	Per comp.
Effectiveness + Introduction	147	177	1.20	0.42	0.0029
Situational leadership	509	531	1.04	0.63	0.0012
Distributed management in education	292	323	1.11	0.53	0.0018
Hoy-Miskal-bridges	204	331	1.62	0.65	0.0032
Lewin forces	263	237	0.90	0.48	0.0018
Other	97	*	*	*	*
Unit total	1512	1599	5.87	2.72	0.0109
Unit mean	302	320	1.06	0.54	0.0018

The average screen took 4.28 *Dh* to produce and was viewed by an average learner for 0.0072 hours (26 seconds). Similarly, the average component took 1.06 *Dh* to produce and was viewed on average for just 0.0018 hours (6.5 seconds). Cates (1994) also found that the average screen was composed of 4.04 components, which consisted of the base screen and 3.04 overlays. In his research Cates (1994) found that correlation analysis suggested that the best predictor of *effort* was the number of screens ($r=+0.917$, $p<0.05$). There was also a strong correlation with the number of components ($r=+0.907$, $p<0.05$). There was a strong correlation between number of screens in a lesson and *learner time* ($r=+0.984$, $p<0.01$) while a strong negative correlation existed between the number of components and the time a learner spent viewing each component ($r=-0.982$, $p<0.01$).

While Jay, Bernstein and Gunderson (1987) rejected screen counts both Cates (1994) and Senbetta (1991) produced similar results for the number of screen counts per learner-hour of *learner time*. A weighted average of the mean reported number of screens suggested by Senbetta's experts produced 127 screens per learner hour while Cates' experiment produced 138 screens per learner hour Cates (1994). These are similar in size and point towards a method for estimating both *learner time* and *effort*.

3.3.3 Interactions

Since the underlying premise of courseware is that students learn by interacting with the computer, the use of interaction as a size measure has been considered by two authors. Yeager (1987) defined an hour of *learner time* as "... approximately 60 [meaningful] interactions". The problem with this size measure is that it requires careful definition of the meaningful interactions. As indicated previously, it is relatively simple to pad out courseware by including an electronic page-turning exercise where the "interaction" consists of "pressing the spacebar" or "point and click". However, Yeager (1987) defined a meaningful interaction to have a one or more of the following three characteristics:

- It must be related to the lesson objectives
- It must occur within a situation where there is a broad range of expected answers and the student must discriminate between several choices
- There must be specific feedback for each answer including unanticipated responses

Senbetta (1991) defined an interaction to be "... an item that presents a scenario or a question requiring a student to respond using an input device such as a keyboard or mouse". He tried to investigate experts' perceptions of the relationship between *learner time* and simple or complex interactions. Any interaction which accepted and processed no more than four types or forms of user input was defined to be simple while one that accepted between five and ten was defined to be complex. However, as Table 3.32 shows, the results were disappointing with no consensus being reached.

Table 3.32 - Number of interactions per hour of *learner time* (Senbetta 1991)

Number of interactions per learner hour	Simple Interactions		Complex Interactions	
	Number of respondents	%	Number of respondents	%
20	2	8	2	8
30	4	16	7	28
40	3	12	5	20
50	4	16	1	4
60	5	20	2	8
Other	1	4	2	8
No response	6	24	6	24

From these results it would appear that the 60 interactions per learner hour proposed by Yeager (1987) are at the high end of those expected by Senbetta's (1991) experts. The

use of meaningful interactions suffers from being difficult to count and based on Senbetta's (1991) experience, it is difficult to communicate the concept even to experienced developers. It is also a measure which could encourage developers to add superfluous questions to keep the number of interactions high (Jay et al. 1987).

3.3.4 Lessons

Although not widely used, a few developers reported using the type and number of lesson segments required as the basis of a *effort* estimation methods (Wilson 1986). This appears to be an extremely arbitrary measurement since developers can define a lesson in almost any number of ways and it is difficult to estimate before the start of the design phase (Jay et al. 1987). Jay, Bernstein and Gunderson (1987) found only 9% of developers claimed to use lessons as the basis for estimation while it was stated by only (3%) of respondents as their preference for an industry standard.

3.3.5 Performance objectives

Objectives have been proposed by a number of authors as the basis for a size measure. However, it shares most of the problems associated with previous measures. While not dependent on *learner time*, there is no evidence that it is directly related to development *effort*. Jay, Bernstein and Gunderson (1987) indicated an *effort* estimation method was being developed by the US Army Training Support Centre (ATSC) which used performance objectives as the basic size measure rather than *learner time*. Performance objectives can only be used once the courseware has been specified. Senbetta (1991) found that 67% of respondents used between 3 and 8 objectives for each hour of *learner time*. Table 3.33 shows the number of objectives required to produce one hour of *learner time*.

Table 3.33 - Number of objectives per hour of *learner time* (Senbetta 1991)

Number of objectives per learner hour	Number of respondents	%
3-5	7	28
6-8	12	48
9-11	4	16
12-15	0	0
No response	2	8
Total	25	100

3.3.6 Media objects

The dominance of media, especially video, has the potential to dominate all other aspects of courseware development. However, there is no empirical evidence to support the use of media objects as a size measure.

3.3.7 Composite courseware measures

Jay, Bernstein and Gunderson (1987) found that 42% of respondents in their survey made use of composite measures to estimate courseware size. The respondents mainly described methods which used combinations of those previously described with the addition of content complexity. In the survey 36% of respondents recommended the use of a composite measure as an industry standard.

3.3.8 Software-based

In the survey presented in Section 3.1 of this Chapter thirty-six courseware developers were asked to identify from a range of alternative types of authoring tool the percentage contribution of each tool to overall courseware development. Authoring languages and authoring systems were identified as the main contributors to development *effort*. While software-based size measures are well established in software *effort* estimation, it is unknown in courseware *effort* estimation. No empirical or other studies could be found which related aspects of the underlying software code or derived measures to courseware *effort*. This is hardly surprising because most authoring tools de-emphasise the programming aspects.

For most authoring systems it is impossible to gain direct access to the underlying code. Early authoring system interfaces described the courseware in terms of linked frames or screens. Many recent authoring systems replace "frame" with "object" but still allow limited or no access to program codes. The other main type of authoring tools are

described as authoring languages. These allow direct access to specialist programming or scripting languages which incorporate features and facilities to assist in the creation of courseware.

Some of the size measures found in software engineering literature may be useful in the analysis of courseware *effort* estimation. The following broad headings will be used to define software-based size measures in terms of the measurement baseline.

- Code-based
- Derived measures

Under each of these headings software-based size measures will be described along with the problems associated with their use in measuring courseware size.

Code-based

Code-based measures attempt to map some measurable aspect of the source code to the size of the finished program. Source lines of code (SLOC) is probably the oldest and most widely used code-based measure of software size (Conte et al. 1986). However, others do exist, such as token counts, graph theoretical measures, and hybrid measures (Shepperd and Ince 1993). The following paragraphs will explore the definition of source lines of code. While these measures are of interest, they are outside the scope of the current research.

Source lines of code (SLOC) is potentially useful with courseware developed using assembly languages, general purpose languages and authoring languages which all allowed direct access to the source code for measurement purposes. However, despite its longevity there are a number of well-documented problems associated with defining and counting source lines of code (Boehm et al. 1994). To minimise these problems the Software Engineering Institute's (SEI) checklist for logical source statements should be used to define what is being counted for each language (Park 1992). Checklists already exist for a number of languages, such as C, C++ and Pascal. The logical source code statement check list should be completed for any authoring language used in courseware development.

However, even with the use of these checklists, there are disagreements among authors as to what to count, especially in the area of code reuse (Park 1992; Boehm et al. 1994). In terms of the courseware measurement baseline all new code developed is included, as is all reused code no matter its source. However, the original *effort* associated with creation of the reusable code is not counted in the total *effort* of each courseware module in which the code is used. Only the *effort* associated with finding, learning, patching, modifying or testing the reused code is measured in each courseware module in which reusable code is used. The aim again is to localise the *effort* expended within a particular courseware module. The existence of reusable templates, tools or courseware modules will be dealt with through the use of an appropriate sizing factor.

While it is relatively simple to automate the collection of source lines of code (Boehm et al. 1994), considerable doubt exists as to the validity of using a size measure which may account for less than 24% of the total project *effort* as the basis for a courseware *effort* estimation model (Marshall et al. 1996). As previously indicated, the design and development of media aspects can dwarf the programming *effort*.

Derived measures

The difficulties associated with using estimates of source lines of code to estimate software *effort* encouraged a number of authors to explore derived measures (DeMarco 1984; Kafura and Canning 1985; DeMarco 1989; Heemstra 1990; Banker et al. 1992; Chidamber and Kemerer 1994; Jones 1995). Function points, bang measures, object points and numerous other derived measures of software size could all be of relevance in a situation where the courseware authoring tool has no source lines of code to measure.

However, none of the existing derived measures of software size were originally designed for multimedia courseware. Most were originally designed for an information management or data processing environment. It is unlikely that they will translate directly into a multimedia courseware development environment without considerable redefinition, calibration and validation. While this is worthy of further investigation, it is outside the scope of this thesis.

3.4 OTHER MEASURES

In this Section the results of a literature review into the proposed output measures will be discussed. Limitations in the published research will be identified along with key information which will form the basis for the proposed paradigm. Such factors will be presented under the following headings:

- Elapsed time
- Development time
- Effort
- Team size
- Full-time equivalent developers
- Cost
- Productivity
- Effort to learner time ratio

These definitions extend the brief descriptions provided in Chapter 3 for the discussion of existing multimedia courseware effort estimation methods.

3.4.1 Elapsed time

Having defined the development life-cycle and agreed the starting and finishing points, *elapsed time* can be formally defined. Based on the earlier informal definition, elapsed time measures the total time in *days (D)* from the start of the instructional design phase to the end of the testing phase. Unfortunately, *elapsed time* is not generally reported in the literature related to courseware development or effort estimation. The only recent reference to *elapsed time* was made indirectly by (Cates 1994) who stated that "... design work took place in an intensive seven week period ... authoring occurred in a nine week period ... also an eight week period of revision and polishing following the initial completion of the five lesson unit." The resulting courseware development had an approximate *elapsed time* of one hundred and sixty eight days. It is a great pity, given the volume of courseware development in the last thirty years, that more developers did not take the opportunity to state in the literature the *elapsed time*. Even if it is informally presented as in Cates (1994), the result is to place the development in a human time-scale.

3.4.2 Development time

Development time is the basic measure of the number of *hours (H)* taken to develop the multimedia courseware from the start of the instructional development phase to the end of the testing phase. During this period *development time* is measured in working *hours* elapsed during the multimedia courseware development excluding weekends, holidays and lunch breaks. *Development time* can be related to *elapsed time* using the following equation.

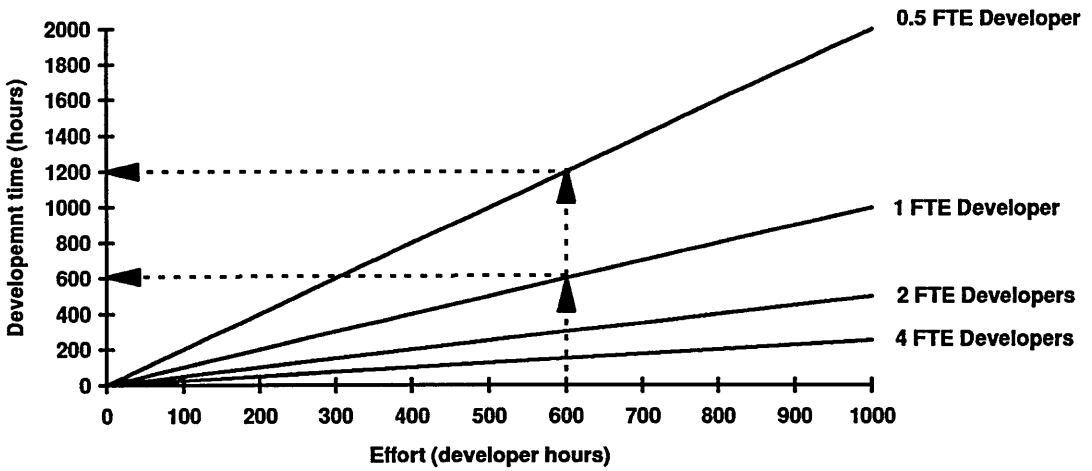
$$Development\ time = (Working\ hours\ per\ day \times (Elapsed\ time - (Non\ working\ days)))$$

Development time appears to be used extensively in the literature (Gagne and Briggs 1979; Jay et al. 1987; Schooley 1988; Senbetta 1991; Golas 1993; Orey et al. 1994; Navassardian et al. 1995). Unfortunately, its use is based on a fundamental misunderstanding over what is being measured. In the studies reviewed, the authors were effectively measuring effort which they use interchangeably with development time. This is a fundamental misunderstanding of the relationship between effort, *development time* and the number of developers involved. The following equation states the relationship.

$$Development\ time = \frac{Effort}{FTE\ Developers}$$

In the literature there is an unstated assumption that only one full-time equivalent (*FTE*) developer is involved. Thus *effort* and *development time* have the same value but are not measuring the same quantity. As Figure 3.13 shows, with only one FTE developer, 600 *Dh* of effort does translate into 600 H of *development time*. However, if the developer was working 50% on this project and 50% on another i.e. 0.5 FTE developer, the 600 *Dh* translates into 1200 H of *development time*.

Figure 3.13 - Graph of development time to effort



3.4.3 Effort

Effort is used to measure the productive work expended to complete a courseware development project. It is calculated over the measurement baseline for valid *development activities* in units of *developer-hours (Dh)*. Most of the studies originally reported results measured using some form of *development time* (hours, days, or months). To remain consistent with the proposed paradigm, all references to *development time* were replaced by effort except where there is clear evidence to support the use of the original term.

The following equation can be used to calculate the total effort expended developing the course from the *development time* and *FTE developers*.

$$Effort = Development\ time \times FTE\ developers$$

The total effort across the life-cycle can be calculated using the following equation.

$$Effort = \sum_{i=1}^{n} Effort_i$$

Where $Effort_i$ is the effort expended in the ith phase of the development life-cycle. However, measuring effort on its own is of very little value as the following two empirical studies show. One of the earliest experimental studies into the courseware

development effort was carried out by Gailey (1973b). In this study, the effort expended by Gailey, five courseware consultants and four subject matter experts to develop four courseware modules was reported. The modules were developed using a well-defined Instructional Systems Development (ISD) model by the small team on a part-time basis. Table 3.34 shows the total development effort by module along with post development refinement of the "Household Electricity" course.

Table 3.34 - Household electricity development effort (Gailey 1973b)

Module/Activity	Instructional Method	[Effort (*Dh*)]
1. Electricity terms	Tutorial	185.51
2. Safe use of electricity	Tutorial	88.00
3. Household circuit game	Simulation game	135.59
4. Cost of using electricity	Tutorial	32.30
Module development total	-	441.40
Refinement after operation	-	50.25
Total	-	491.65

More recently Cates (1994) presented the results of an empirical investigation of the development *effort* required by novices to produce five courseware lessons in educational management topics.

Table 3.35 - Development *effort* data (Cates 1994)

Lesson Name	[*Effort* (*Dh*)]	[Mean *Learner time* (*Lh*)]
Effectiveness + Introduction	177	0.42
Situational leadership	531	0.63
Distributed management in education	323	0.53
Hoy-Miskal-bridges	331	0.65
Lewin forces	237	0.48
Total	1599	2.71
Unit mean	320	0.54

As Table 3.34 and Table 3.35 show, it is difficult to draw meaningful conclusions from two experiments without an appropriate size measure to allow comparison of the coursewares developed.

3.4.4 Team size

The team size measure describes the total number of developers actively involved in the courseware project team. Studies which mention developers tend to do it in passing and only give total team size involved in the project. For example, Cates (1994) indicated that the "... design team consisted of eleven novice designers and the author. ... [while the] ...authoring team consisted of five novice authors and the writer." There is also

little guidance from the literature regarding who to include or exclude as a member of the development team. However, the paradigm definition of development activities does serve as a basis by which to identify team members. *Team size* is defined to be the total number of people involved in the project undertaking *development activities*. Assuming that all the developers in Cates' (1994) example were involved in valid *development activities,* the *team size* is seventeen. In simple terms *team size* is a raw count of the total number of developers involved in the project.

3.4.5 Full-time equivalent developers

In the Cates' (1994) study there is no indication of the developers' commitment to the project. Were they working full-time or part-time on this project? In this case the developers were students, so it is reasonable to assume they were working on other studies at the same time. If all seventeen developers in the team were working part-time on the project, how does this compare with full-time equivalent (FTE) developers? In this particular example it is very difficult to tell because Cates (1994) did not publish the data. However, it does not remove the need to have a measure which describes the number of developers involved independent of the project-staffing conditions. *Full-time equivalent developers (FTE)* attempts to do this by converting the contribution to *effort* by full-time, part-time and contract developers into the equivalent number working full-time on the project.

Again, the definition for *development activity* is used to identify which developers count towards this measure. The following equation describes how to calculate the number of *FTE* developers involved in the project.

$$Full\ time\ equivalent\ developers = \sum_{i=1}^{Team\ size} \frac{Effort_i}{Development\ time_i}$$

Where $Effort_i$ is the *effort* expended by the *i*th developer in the team over the $Development\ time_i$ for valid *development activities*. The contribution of each developer in the team is summed from one to *team size*. The *resulting Full time equivalent developer* measure is measured in *FTE developer* units.

3.4.6 Cost

In the literature there is a dearth of experimental and other studies into the *cost* of developing courseware. The literature that does exist tends to be more anecdotal or survey-based. There also appears to be some confusion over what is being measured. Some authors use *cost* to describe the payroll *costs* of developer *effort* while others use it to encompass payroll and non-payroll costs. As defined previously, *cost* in the measurement paradigm describes the monetary value of the *effort* expended to undertake valid *development activities* and other overhead *costs* incurred during this period.

3.4.7 Productivity

Symons (1993) defined a range of productivity measures for software size estimation. The basic measure of productivity is given by dividing the output by the input. The output measure is the *learner time* delivered and the input is the *effort*. Productivity is thus measured by the following equation.

$$Productivity = \frac{Learner\ time}{Effort}$$

Productivity is measured in units of *mean learner hours per developer-hour* ($\overline{Lh^{-1}}\ Dh$). With current development methods, productivity will normally result in a value less than one.

As defined in the introduction to this Chapter, no reference to productivity could be found in the literature related to courseware development.

3.4.8 Effort to learner time ratio

The preferred method of measuring courseware productivity is the *effort to learner time ratio*. The following equation can be used to calculate *the effort to learner time ratio*.

$$Effort\ to\ learner\ time\ ratio = \frac{Effort}{Learner\ time}$$

Using this definition, the ratio is measured *in developer hours per mean learner hour*

($Dh\ \overline{Lh^{-1}}$). Inverting the equation for *effort to learner time ratio* produces the following equation for *productivity*.

$$\text{Effort to learner time ratio} = \frac{1}{\text{Productivity}}$$

Table 3.36 shows examples of the relationship between *productivity* and *effort to learner time ratios*.

Table 3.36 - Comparison of *productivity* and *effort to learner time ratios*

Learner time ***(Lh)***	***Effort*** ***(Dh)***	***Productivity*** ($Dh\ \overline{Lh^{-1}}$)	***Effort to learner time ratio*** ($Dh\ \overline{Lh^{-1}}$)
0.5	100	0.005	200
1	100	0.01	100
2	100	0.02	50
20	100	0.2	5
100	100	1	1

As Table 3.36 shows, the *effort to learner time ratios* are easier to remember than the proposed *productivity* equivalent and it is likely to remain popular due to its historical significance and acceptability to industry. However, the use of *productivity* will ensure compatibility with the equivalent software measure.

3.5 EVALUATING THE MEASUREMENT PARADIGM

Previous Sections outlined a measurement paradigm for courseware effort estimation models. Although superior to the existing situation in courseware effort estimation, the paradigm has a number of easily identifiable weaknesses. In this Section these weaknesses will be explored before describing a checklist to assist with data collection and evaluation of effort estimation models.

3.5.1 The measurement paradigm

Defining a measurement paradigm to describe multimedia courseware development will allow a more objective comparison to be made of productivity and will form the basis for a multimedia courseware effort estimation model. Defining the organisational context and end product quality of multimedia courseware ensures that at least the measurements are likely to produce similar values in comparable circumstances. The measurement paradigm has attempted to clarify what is being measured and how it is measured. However, it still suffers from a number of weaknesses concerning its definition.

Measuring courseware quality

The proposed paradigm makes use of deliverables to try and establish a common measurement of courseware end product quality. This is, however, not totally acceptable in terms of a product which has educational or training objectives. It is entirely possible to expend considerable effort producing courseware with all the deliverables mentioned which nevertheless does not produce the educational result expected. The two measures proposed as indicators of educational quality, while trying to overcome this problem, are not perfect and may not measure educational quality in every situation. Defining a universal measurement of courseware quality which everyone agrees on is not a simple task and this area requires further research. It is even harder to define predictive measures of quality which could be useful in development effort estimation models but this is in any case beyond the scope of this thesis.

Use of the waterfall model

The waterfall model for multimedia courseware development, while highly stylised, allows the starting and finishing points for productivity and cost estimation

measurement purposes to be determined. Based on this, consistent definitions of *elapsed time*, *development time*, *effort*, *productivity* and *effort to learner time ratio* can be used. However, the waterfall model is seen by some as outdated having been replaced by a rapid prototyping model in both software and courseware engineering. The function of the waterfall model in the paradigm is to indicate the start and the end points for measurement. Whatever model is used an equivalent of the waterfall model courseware specification and testing phases can be defined. For example, in a rapid prototyping model an initial courseware specification has to be agreed with the client prior to prototyping and the product is defined to be finished at some point. It should be a relatively simple task to re-map the measurement baseline to an alternative software or courseware engineering model.

3.5.2 Size measures

Size measures are still the weakest aspect of research into courseware effort estimation models. Each of the alternative measures suffers from the potential variations caused by inaccurate measurement or estimation. The following paragraph shows an example of the potential problems using *learner time* although other measures experience similar problems.

The effect of learner time

Apparent improvements in *productivity* can be made by using pessimistic values of *learner time*. Table 3.37 gives an indication of the effect on *productivity* of using lowest, mean and highest values of *learner time*.

Table 3.37 - Effect of different values of *effort* and *learner time* on *productivity*

Effort (*Dh*)	Productivity ($\overline{Dh\ Lh^{-1}}$)		
	Calculated using the lowest value of *Learner Time* (1.44 *Lh*)	Calculated using the mean value of *Learner Time* (3 *Lh*)	Calculated using the highest value of *Learner Time* (5.46 *Lh*)
100	0.0144	0.0300	0.0546
200	0.0072	0.0150	0.0273
300	0.0048	0.0100	0.0182
400	0.0036	0.0075	0.0136

From these results it is clear that great care must be taken in measuring *learner time* to prevent inaccurate claims for productivity improvement. The values of *learner time* are based on Gailey's (1973a) results for students using her "Home Economics"

courseware.

Software-based measures

Software-based measures all suffer from the problem that there has been no published research into the relationship between courseware development effort and code-based measures. It is doubtful given the dominance of media in courseware that there is a simple relationship between effort and code-based measures. One aspect of this current research is to explore this area.

3.5.3 Productivity

The formal definition of the measures and their associated units has helped to clarify a number of areas of interest in the measurement of courseware development effort. The consistent use of units in the previous Chapter helped to highlight weaknesses of and misunderstandings about the relationships that exist between the different aspects of *effort*, *development time* and *productivity*. However, despite the clarification provided by this measurement paradigm, there are still problems concerning the inaccurate measurement of the development life-cycle. When this is combined with the possible variation resulting from inaccurate estimation of *learner time* the effect on the range of potential *productivity* is extremely large.

The effect of phase exclusion on effort and productivity

Agreeing the starting and finishing points in a development life-cycle is important because excluding phases has an impact on effort to produce multimedia courseware. Table 3.38 indicates the effect of excluding life-cycle phases on effort and on the resulting productivity for a nominal 3 *Lh* of *learner time*.

Table 3.38 - Effect of excluding life-cycle phases on effort

Phase	*Effort (Dh)*			
	%	Project A	Project B	Project C
Courseware Specification	7	21		
Instructional Design	34	102	102	
Multimedia Design	25	75	75	75
Multimedia Development	22	66	66	66
Courseware Integration	8	24	24	24
Testing	4	12	12	
Total	100	300	279	165
Effort (% of Measurement baseline)		108%	100%	59%
Productivity ($\overline{Lh}\ Dh^{-1}$)		0.0100	0.0108	0.0182

The percentage allocated to each phase of the life-cycle is based on figures quoted by Imke (1991) for courseware development. The effort quoted for each phase is based on these percentages for a nominal 300 *Dh* project measured using all development phases in the life-cycle. Project A describes a development in which all phases in the life-cycle, excluding maintenance, are recorded. This increases the recorded effort to 108% of that using the measurement baseline described by Project B. Project B uses the agreed multimedia courseware measurement baseline in which the effort for Courseware Specification is excluded. In Project C the Instructional Design and Testing phases are also removed as may happen when an existing course is re-developed using another authoring system. The result is to reduce effort to 59% of that for Project B. Thus, by not recording all phases in the development life-cycle, it is possible to reduce the apparent effort and almost double the *productivity* value without any real improvement in productivity.

Combined effects of variation in learner time and phase exclusion

Table 3.39 indicates the combined effect on *productivity* using the variation in *learner time* and effort due to life-cycle phase exclusion.

Table 3.39 - Combined effect on productivity of variation in *effort* and *learner time*

Learner Time	Productivity ($\overline{Lh}\ Dh^{-1}$)		
	Project A (300 *Dh*)	Project B (279 *Dh*)	Project C (165 *Dh*)
Low (1.44 Lh)	0.0048	0.0052	0.0087
Mean (3 Lh)	0.0100	0.0108	0.0182
High (5.46 Lh)	0.0182	0.0196	0.0331

Table 3.39 helps to explaining the difficulty experienced in measuring both *productivity* and effort estimation for courseware development. Even with a rigorously defined measurement paradigm, it is possible to produce a range in which the highest value is 6.90 times the lowest value of *productivity*. This is consistent with the range of values produced by Senbetta's (1991) expert estimators for a well defined courseware specification. While measurement of the effort can be controlled by consistent definitions and measurements, the normal variation in *learner time* is inevitable and has a greater potential to produce inconsistent results when used to measure *productivity* or as the basis of courseware effort estimation models.

3.5.4 Measurement paradigm checklist

To assist with the collection of courseware development effort data there is a need to ensure that the standards defined in the measurement paradigm are used. Table 3.40 describes a checklist based on the measurement paradigm. This will be used in Chapter 6 to compare the situations in which data was collected in each of the case studies.

Table 3.40 - Measurement paradigm checklist for evaluating effort estimation models

Heading	Sub-heading	Description	Yes	No
Organisational context	Commercial	Sole author		
		Team		
		Consortium		
	Non-commercial	Sole author		
		Team		
		Consortium		
Courseware product quality	Status	Finished testing phase and ready for publishing		
		Produced to master copy or camera ready quality		
	Includes	Software which controls the learning environment		
		Computer-based presentation of one or more media		
		Computer-based assessment of learning		
		Learning support materials		
		Installation materials		
		Learner record system		
Educational quality	Mastery score gain	Mean greater than 50%		
	Completion rate	Greater than 95%		
Measurement baseline	Includes	Instructional design phase		
		Multimedia design phase		
		Multimedia development phase		
		Courseware integration phase		
		Testing phase		
	Excludes	Courseware specification phase		
		Maintenance phase		
Development activities (within measurement baseline)	Included *effort*	Development team members		
		Subject matter experts based with client		
	Excluded *effort*	Client's employees/learners		
		Non-team employees of developer		
Sizing factors	Learner time	Mean measured from a sample of 30 target learners		
		Estimated by client/developer		
	Screens	Base screens and overlays		
	Interactions	Meaningful and related to course objectives		
	Lessons	Discrete/named subsection of course		
	Objectives	Number of specific objectives		
	Media objects	Number of discrete media objects		
	Source lines of code	Logical source lines of code		
Measures	Elapsed time	Measurement baseline measured in days		
	Development time	Measurement baseline measured in working hours		
	Effort	Measurement baseline *effort* measured in *Dh*		
	Team size	Total number of developers on the team		
	FTE developers	Full-time equivalent developers on the team		
	Cost	Monetary value of effort and overheads		
	Productivity	Output/input measured in $\overline{Lh\ Dh^{-1}}$		
	ETLTR	*Effort* to learner time ratio measured in $\overline{Dh\ Lh^{-1}}$		
	DEC	Developer *effort* charge to client (Currency Dh^{-1})		

3.6 SUMMARY

The main finding of this survey was that the use of a rigorous estimation method increased the reported accuracy of courseware 'cost' estimates. There was a tendency for respondents to describe courseware "cost" in terms of either monetary value or developer effort. In terms of sizing factors most respondents reported using either expected *learner hours* or number of screens. Five potential productivity adjustment factors were used to analyse the results of the survey. There were no significant differences among most of the productivity adjustment factors and the apart from use of a rigorous estimation method. The mean reported accuracy of the estimates produced were within 20% for commercial and 41% for non-commercial developers.

The effort to learner time ratio has come to reign supreme as the cornerstone measure of courseware development productivity. Unfortunately, in both *productivity* measurement and more particularly, courseware effort estimation, the results produced by this measure have not been reliable. However, even with this paradigm, the effect of inconsistent measurements of development *effort* and *learner time* can produce extreme variations in both *productivity* or *effort to learner time ratios*. Any measure based on *learner time* is subject to optimistic or pessimistic measurement or, worse still, estimation and as such, its value in either comparing *productivity* or as the basis for an effort estimation model is limited. A more stable measure is required especially for the development of effort estimation models and this forms an important part of this research. The measurement paradigm checklist provides a method of ensuring consistency in data collection and comparing the results of effort estimation models.

The next Chapter examines the use of productivity adjustment factors in effort estimation modelling.

3.7 REFERENCES

Avner, A., 1994. *Personal communication.* (E-mail to I.M. Marshall, 10 July 1994) Urban-Champaign, IL: University of Illinois, Computer Based Education Research Laboratory.

Avner, A., Smith, S. and Tenczar, P., 1984. CBI authoring tools: effects on productivity and quality. *Journal of Computer-Based Instruction.* **11**: pp. 85-89.

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

Barker, P. and King, T., 1994. Evaluating interactive multimedia courseware - a methodology. *Computers and Education.* **21(4)**: pp. 307-319.

Bennet, A.J., 1990. Objectively determining the educational potential of computer and video-based courseware; or producing reliable evaluations despite the dog and pony show. In: *7th International Conference on Technology and Education, Brussels, Belgium, 20-22 March.* p. 18.

Bergman, R.E. and Moore, T.V., 1990. *Managing interactive video/multimedia projects.* Englewood Cliffs, NJ: Educational Technology Publications.

Boehm, B. et al., 1994. *Cost models for future software life cycle processes: COCOMO 2.0.* (Annals of Software Engineering Draft 2.1), 21 September 1994, Los Angeles, CA: University of Southern California.

Canale, R. and Wills, S., 1995. Producing professional multimedia: project management issues. *British Journal of Educational Technology.* **26(2)**: pp. 84-93.

Cates, W.M., 1994. Estimating the time required to produce computer-based instructional lessons: Descriptive analyses of the production data of novice instructional developers. *Journal of Educational Computing Research.* **10(1)**: pp. 29-40.

Chidamber, S.R. and Kemerer, C.F., 1994. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*. **20(6)**: pp. 476-493.

Conte, S.D., Shen, V.Y. and Dunsmore, H.E., 1986. *Software engineering metrics and models*. Menlo Park: Benjamin/Cummings Publishing Company Inc.

DeMarco, T., 1984. An algorithm for sizing software products. *Performance Evaluation Review*. **12(2)**: pp. 13-22.

DeMarco, T., 1989. In the land of function metrics. In: *5th International COCOMO Users' Groups Conference, Pittsburgh, PA,*

Fairweather, P. and O'Neal, A., 1984. The impact of advanced authoring systems on CAI productivity. *Journal of Computer-based Instruction*. **11**: pp. 90-94.

Gagne, R.M. and Briggs, L.J., 1979. *Principles of instructional design*. 2nd Edition ed. New York: Holt, Rinehart & Winston.

Gailey, F., 1973a. An analysis of development: Use time ratios for a computer assisted instruction unit on basic household electricity. In: *Association for the Development of Computer-based Instructional Systems, Ann Arbor, Michigan, 7-9 August 1973*. pp. 1-8.

Gailey, F.H., 1973b. *An analysis of development/use time for a computer-assisted instruction unit on basic household electricity*. (PhD thesis, Ohio State University).

Gery, G., 1987. *Making CBT happen: Prescriptions for successful implementation of computer-based training in your organisation*. Boston, MA: Weingarten Publications Inc.

Golas, K.C., 1993. Estimating time to develop interactive courseware in the 1990s. In: *13th Interservices Industry Training and Education Conference, Orlando, FL, November 1990*.

Heemstra, F.J., 1990. Software cost estimation model. In: *5th Jerusalem Conference on Information Technology, Jerusalem, Israel,* pp. 286-297.

Hellman, M.F. and James, W.R., 1995. *The multimedia casebook.* New York: Van Nostrand Reinhold.

Hirschbuhl, J.J., 1989. CBT Surveys for Computer Based Training Directions. *Interactive Learning International.* **5**: pp. 55-70.

Hobbs, P. and Price, S., 1994. Educational project management; Repeat after me. In: *ALTC 94, Enabling Active Learning,, University of Hull, Hull, UK, 18-21 September 1994.*

Imke, S., 1991. *Interactive video management and production.* Englewood Cliffs, NJ: Educational Technology Publications.

Jay, J., Bernstein, K. and Gunderson, S., 1987. *Estimating computer-based training development times.* ARI Technical Report (765), October 1987, Alexandria, VA: Research Institute for Behavioural and Social Sciences.

Jones, C., 1995. Measurement in practice: the expanding role of function point metrics. In: *European Software Cost Measurement Conference, Rolduc, Netherlands, 17-19 May 1995.* pp. 5.1-5.31.

Kafura, D. and Canning, J., 1985. A validation of software metrics using bang metrics and two resources. In: *8th International Conference on Software Engineering,* pp. 378-385.

Kearsley, G., 1985. The CBT advisor: An expert system program for making decisions about CBT. *Performance and Instruction.* **24(9)**: pp. 15-17.

Kitchenham, B., Pfleeger, S.L. and Fenton, N., 1995. Towards a framework for software measurement validation. *IEEE Transactions on Software Engineering.* **21(12)**: pp. 929-944.

Kulik, C.C. and Kulik, J.A., 1986. Effects of computer-based education in colleges. *Association for Educational Data Systems Journal.* **19**: pp. 81-108.

Lisewski, B. and Settle, C., 1995. Teaching with multimedia: a case study in weed biology. *Active Learning*. **3(December)**: pp. 28-35.

Lord Kelvin, 1889. Popular lectures and addresses. In: Quoted in: Cook, M.L., 1982. Software metrics: an introduction and annotated bibliography. *ACM SIGSOFT Software Engineering Notes*. **7(2)**: pp. 41-60.

Marshall, I.M. et al., 1996. Analysis of expert opinion on multimedia courseware effort estimation using cost drivers. **In press.**

Marshall, I.M., Samson, W.B. and Dugard, P.I., 1994a. Estimating multimedia development effort for open access education. In: *AETT 94 Computer Assisted and Open Access Education, Edinburgh, UK,*

Marshall, I.M., Samson, W.B. and Dugard, P.I., 1994b. Multimedia courseware cost modelling. In: *European Software Cost Modelling Conference 94, Ivera, Northern Italy, 11-13 May*. pp. 27.21-27.19.

Marshall, I.M., Samson, W.B. and Dugard, P.I., 1994c. Predicting the development effort and cost of multimedia courseware for open access education. In: *29th International Conference of the Association for Educational Technology, Napier University, Edinburgh, 11-13 April.*

Marshall, I.M. et al., 1995a. A rigorous framework for measuring development productivity and estimating the cost of multimedia courseware. In: *Sixth IFIP World Conference Computers in Education, Birmingham, UK, 23-28 July*. pp. 139-149.

Marshall, I.M. et al., 1995b. The mythical development to learner time ratio. *Computers and Education*. **25(3)**: pp. 113-122.

Marshall, I.M. et al., 1994d. Predicting the development effort of multimedia courseware. *Information and Software Technology*. **36(5)**: pp. 251-258.

Marshall, I.M. et al., 1995c. Evaluating multimedia courseware cost models. In: *European Software Cost Measurement Conference, Rolduc, Netherlands, 17-19 May 1995*. pp. 6.1-6.14.

Mikos, R. et al., 1987. Estimating training effort/cost: A follow-up 'final report' (Part 1). *Performance & Instruction.* **26(5)**: pp. 24-29.

Miles, K., 1990. A generic model for rapid estimation of CBT development time. In: *13th InterAgency/Industry Training Systems Conference, Orlando, FL, November 1990.*

Miles, K.W. and Griffith, E.R., 1993. Developing an hour of CBT: The quick and dirty estimate. *CBT Directions.* **(April-May)**: pp. 28-33.

Milette, M. and Trevor-Deutsch, L., 1995. *Training cost calculation spreadsheet: classroom versus computer-based training.* Visual Solutions Inc.

Navassardian, S., Marinov, M. and Pavlova, R., 1995. Investigation on the quality and efficiency of instructive computer-aided training. *British Journal of Educational Technology.* **26(2)**: pp. 109-121.

Orey, M. et al., 1994. Development efficiency and effectiveness of alternative platforms for intelligent tutoring for the mobile subscriber radio-telephone terminal. *Computers and Education.* **22(4)**: pp. 301-313.

Orlansky, J. and String, J., 1979. *Cost-effectiveness of computer-based instruction in military training.* Technical Report (P-1375), April 1979, Institute for Defence Analysis.

Park, R.W., 1992. *Software size measurement: a framework for counting source statements.* Technical Report (CMU-SEI-92-TR-20), Pittsburg, PA: Software Engineering Institute, Carnagie Mellon University.

Philips Professional Media, 1996. *Costing a production.* UK: Philips Media Professional.

Reeves, T.C., 1991. Ten commandments for the evaluation of interactive multimedia in higher education. *Journal of Computing in Higher Education.* **2(2)**: pp. 84-113.

Reeves, T.C., 1992. Evaluating interactive multimedia. *Educational Technology.* **May**: pp. 47-53.

Roblyer, M.D., Castine, W.H. and King, F.J., 1988. *Assessing the impact of computer-based instruction.* New York: Hayworth Press.

Ross, S.M. and Morrison, G.R., 1993. Evaluation as a tool for research and development: Issues and trends in IT applications in educational technology. In: *NATO ASI Automating, Instructional Design, Development and Delivery, Grimstad, Norway,*

Schooley, R.E., 1988. Computer-based training (CBT) cost estimating algorithm for courseware (CEAC). In: *11th Interservice Industry Training Systems Conference,* pp. 319-328.

Senbetta, G., 1991. *An inquiry of time and cost estimating for computer-based training courseware design and development as determined by modified Delphi method.* (Ph.D. thesis, Purdue University).

Shepperd, M. and Ince, D., 1993. *Derivation and validation of software metrics.* Oxford: Oxford University Press.

Soloman, M.B., 1994. What's wrong with multimedia in higher education? *Technological Horizons in Education Journal.* **21(7)**: pp. 81-83.

Symons, C.R., 1993. *Software sizing and estimating: Mk II FPA (function point analysis).* Chichester: John Wiley & Sons.

Tan, W. and Nguyen, A., 1993. *Lifecycle costing models for interactive multimedia systems: Interactive multimedia practice and promise.* London: Kogan Page.

Tennyson, R.D., 1995. Instructional system development: the fourth generation. In: R.D. Tennyson and A.E. Barron, Editors *Automating instructional design computer-based development and delivery tools.* Berlin: Springer-Verlag, pp. 33-78.

Tennyson, R.D. et al., 1995. Employment of system dynamics in modelling of instructional design (ISD). In: R.D. Tennyson and A.E. Barron, Editors *Automating instructional design: computer-based development and delivery tools.* Berlin: Springer-Verlag, pp. 603-609.

Vogt, E., 1995. *Personal communication.* (Verbal to M.F. Hellman,) San Diego, California: Black Cat Interactive Design. Quoted in: Hellman, M.F. and James, W.R., 1995. *The multimedia casebook.* New York: Van Nostrand Reinhold.

Wilson, L., 1986. *Personal communication.* (Letter to J. Jay, September 1986) USA: Ford Areospace and Communication Corrporation. Quoted in: Jay, J., Bernstein, K. and Gunderson, S., 1987. *Estimating computer-based training development times.* ARI Technical Report (765), October 1987, Alexandria, VA: Research Institute for Behavioural and Social Sciences.

Yeager, R., 1987. *Personal communication.* (Letter to J. Jay, January 1987) USA: Intercom Inc. Quoted in: Jay, J., Bernstein, K. and Gunderson, S., 1987. *Estimating computer-based training development times.* ARI Technical Report (765), October 1987, Alexandria, VA: Research Institute for Behavioural and Social Sciences.

Zimmerman, K., 1988. Computer and video-based training officers consistencies and savings. *Bank Systems & Equipment.* **October**: pp. 82-85.

3.8 ENDNOTES

[1] Jay, Bernstein and Gunderson (1987) use the term experienced to describe respondents who reported working in courseware development for three or more years. Respondents with between zero and two years were classified as inexperienced.

[2] Jay, Bernstein and Gunderson (1987) describe respondents as accurate estimators if they reported being able to estimate within 10% of the actual development *cost.* Other respondents were classified as inaccurate estimators.

[3] Senbetta (1991) used "One hour of CBT" as the basis of his study.

[4] Gailey (1973a; 1973b) measured the courseware "development time" in hours and "student terminal time" in "minutes". The results are presented in terms of "development to average student use time ratios". The paradigm terms for *effort, learner time, development time* and *effort to learner time ratios* are substituted.

[5] Lisewski and Settle (1995) measured student usage in "student hours" learner time is substituted.

[6] Senbetta (1991) used "reduction in student time" for a course converted from instructor delivery to computer-based training. The values were redefined in terms of courseware compression.

[7] Cates (1994) uses "development time" measured in "hours" rather than *effort* and "on-screen time" measured in "minutes" rather than *learner time* measured in *learner-hours.* The paradigm terms have been substituted.

4. PRODUCTIVITY ADJUSTMENT FACTORS

> "How many hours does it take to develop an hour of [courseware]? I know you need to know the answer, but if I hear that question asked that way one more time, I'll scream. The question frustrates me since there is no straightforward answer. ... In any case, when I do respond to the question, the answer is "It depends". And what it depends on is a number of situational variables that can be evaluated and determined only on a case by case basis." (Gery 1987)

Gery (1987) describes them as "situational variables"; in software effort estimation they tend to be called "cost drivers" (Boehm 1981). In this thesis they will be described as productivity adjustment factors (Kitchenham 1992) to remain consistent with the measurement paradigm and to de-emphasise the implied relationship with cost. Productivity adjustment factors are used by a number of existing courseware effort estimation methods to tailor the resulting effort to the development situation (Kearsley 1985; Gery 1987; Schooley 1988; Golas 1993; Miles and Griffith 1993; Marshall et al. 1994; Milette and Trevor-Deutsch 1995).

In this Chapter productivity adjustment factors found in existing courseware effort estimation methods and other studies will be presented under the following headings.

- What are productivity adjustment factors?
- Course productivity adjustment factors
- Subject matter productivity adjustment factors
- Quality productivity adjustment factors
- Technical productivity adjustment factors
- Client productivity adjustment factors
- Developer productivity adjustment factors
- Identifying the key productivity adjustment factors

The productivity adjustment factors are presented along with, where appropriate, their descriptions, rating scales and supporting courseware development research.

4.1 WHAT ARE PRODUCTIVITY ADJUSTMENT FACTORS?

In software engineering terms productivity adjustment factors are what Boehm (1981) called "cost drivers". Unfortunately, this name implied a relationship with *cost* which is neither accurate nor helpful. The term productivity adjustment factors was chosen because it accurately describes their effect on effort in estimation methods.

4.1.1 Effort estimation methods

All of the existing courseware effort estimation methods are based on the following equation.

$$Effort = Learner\ time \times Productivity\ adjustment\ factors$$

Where the effort is measured in *Dh, learner time* is measured in *Lh* units and the *productivity adjustment factors* are measured in $Dh\ \overline{Lh^{-1}}$ units. In existing effort estimation methods productivity adjustment factors perform two functions. Firstly, they tailor the effort expended to the perceived development conditions and secondly, they act as a dimensionality constant. several effort estimation models based on *learner time* and *productivity adjustment factors* will be discussed in Chapter 5.

In the majority of the methods investigated the productivity adjustment factors are normally described as *effort to learner time ratios*. However, as Bergman and Moore (1990) illustrated in their *'Q' factor* effort estimation method, this is not necessarily the only format for productivity adjustment factors. The following equation describes the productivity adjustment factor used by Bergman and Moore (1990).

$$Productivity\ adjustment\ factors = \frac{Q \times 100000}{Developer\ effort\ charge}$$

If Q is measured in $\$\ Lh^{-1}$ units and the *developer effort charge* is measured in $\$\ Dh^{-1}$ units, the resulting productivity adjustment factor must be measured in $Dh\ \overline{Lh^{-1}}$ units to maintain dimensional consistency. In this case the measurement unit of the productivity adjustment factor must still be measured in $Dh\ \overline{Lh^{-1}}$ because the sizing factor is measured in *Lh* units; only if the underlying sizing factor was changed would the

dimensionality of the associate productivity adjustment factor change.

The existing courseware effort estimation methods are specific examples of the general form of an effort estimation model (Kitchenham 1992; Shepperd et al. 1996). The following equation describes the generalised form of the effort estimation equation.

$$Effort = \alpha(Size)^{\beta}$$

Where *size* is an appropriate measurement of courseware size, α is the productivity adjustment factor coefficient and β is an economy of scale exponent. As indicated previously, the function of α is to tailor the estimate of productivity to the circumstances which exist in the development environment. The economy of scale exponent β in existing courseware effort estimation methods is equal to one. A value of less than one could be used to indicate an economy of scale in which larger projects are more productive than smaller projects. Diseconomy of scale can be indicated by a value greater than one indicating that larger projects are less productive than smaller projects. In addition, α acts as a dimensionality coefficient which converts the units of *size* to the units of effort measured in *Dh*.

4.1.2 Courseware effort estimation studies reviewed

Twelve studies into courseware effort estimation were reviewed to identify the courseware productivity adjustment factors used (Kearsley 1985; Gery 1987; Jay et al. 1987; Mikos et al. 1987b; Schooley 1988; Bergman and Moore 1990; Miles 1990; Senbetta 1991; Golas 1993; Tan and Nguyen 1993; Marshall et al. 1994; Milette and Trevor-Deutsch 1995). Although there was considerable overlap and duplication, references to over five hundred individual productivity adjustment factors were found in the studies reviewed. Table 4.1 shows the letter codes assigned to each of the studies reviewed

Table 4.1 - Letter codes assigned to the twelve studies reviewed

Study	Authors
A	(Jay et al. 1987)
B	(Senbetta 1991)
C	(Gery 1987)
D	(Schooley 1988)
E	(Kearsley 1985)
F	(Miles and Griffith 1993)
G	(Milette and Trevor-Deutsch 1995)
H	(Tan and Nguyen 1993)
I	(Mikos et al. 1987b)
J	(Golas 1993)
K	(Bergman and Moore 1990)
L	(Marshall et al. 1994)

The remainder of this Section summarises the studies reviewed and describes the methods used to identify and group related productivity adjustment factors under suitable headings.

Jay, Bernstein and Gunderson (1987)

This study presented a list of eighty-one productivity adjustment factors[1] classified into seven broad groupings. The following list describes the groupings used along with the number of productivity adjustment factors classified under each.

- Courseware variables (17)
- Technical variables (9)
- Project scope (5)
- Client characteristics (16)
- Developer characteristics (24)
- Quality factors (5)
- Content/audience variables (5)

Jay, Bernstein and Gunderson (1987) drew heavily from the list of productivity adjustment factors described by Gery (1987). This original list was supplemented by productivity adjustment factors suggested by their analysis of questionnaires, interviews and evaluation of Kearsley's (1985) CBT Analyst. In addition, the report mentioned software engineering effort estimation models including COCOMO (Boehm 1981) which appears to have inspired some of the productivity adjustment factors suggested.

While the list of potential productivity adjustment factors was extensive, Jay, Bernstein and Gunderson (1987) tended to present several attributes under one description. For example, a productivity adjustment factor was described as "amount and type of video to be produced". Deciding what was meant by this productivity adjustment factor was not helped by the absence of measurement values, units and scale types or examples to help with classification. In general, productivity adjustment factors identified by Jay, Bernstein and Gunderson (1987) will not be discussed in detail in the following Sections.

Jay, Bernstein and Gunderson (1987) also presented an ordered list of productivity adjustment factors ranked in terms of their effect on effort by the one hundred and seventy nine respondents to their questionnaire. This will be used later in this Chapter to help identify critical productivity adjustment factors.

Senbetta (1991)

In his Delphic study Senbetta (1991) presented twenty five courseware development experts with a list of potential productivity adjustment factors[2]. Twenty-eight productivity adjustment factors were identified under the following three groupings.

- Product factors (9)
- Resource factors (7)
- Client factors (12)

The final list included four additional productivity adjustment factors and seven amendments to original descriptions which were suggested by respondents. Senbetta (1991) made no attempt to fully describe the measurement values, units or scale types. Unfortunately, some of the descriptions cover more than one attribute and, in at least one case, jointly describe both client and developer. In general, Senbetta's (1991) results will not be discussed in terms of individual productivity adjustment factors. However, the ranked ordering of productivity adjustment factors presented by Senbetta (1991) will be used to identify critical productivity adjustment factors later in this Chapter.

Gery (1987)

Gery (1987) presented thirty-six productivity adjustment factors[3] in her informally defined courseware effort estimation method. These were described under the following four broad groupings.

- Courseware variables (14)
- Technical variables (5)
- Human variables (12)
- Other variables (5)

The productivity adjustment factors were rated as high, medium or low in terms of their effect on *development time* and *cost*. Some of the productivity adjustment factor descriptions included examples but, unfortunately, some were incomplete. For example, the high rating may have an example description but not the low or medium values. Even the complete examples are still subjective and could be improved. This weakness will not be discussed further in terms of Gery's (1987) individual productivity adjustment factors presented later in this Chapter.

In common with several other studies reviewed, Gery (1987) had a tendency to describe individual productivity adjustment factors which covered move than one attribute. This, again, made it difficult to analyse and classify the potential productivity adjustment factors into the appropriate grouping. The values shown for Gery's (1987) productivity adjustment factors are measured in units of $Dh\ \overline{Lh^{-1}}$.

Schooley (1988)

In the CEAC effort estimation method Schooley (1988) used eleven productivity adjustment factors[4] described under the following two broad groupings.

- Course independent (2)
- Course dependent (9)

The productivity adjustment factors were well defined with values, units and scales. Schooley (1988) also discussed an example project which assisted with the analytical process. The values shown for Schooley's (1988) productivity adjustment factors are normally dimensionless. They represent the values used in the CEAC equation to

calculate effort.

Kearsley (1985)

Kearsley's (1985) CBT Analyst courseware effort estimation method used twenty-two productivity adjustment factors[5]. Each of the productivity adjustment factors had an associated description, measurement value and scale type. However, as Jay, Bernstein and Gunderson (1987) found in their informal evaluation, most of the scales are not well defined and some were counter intuitive. For example, an unknown rating for a productivity adjustment factor tended to have no effect on productivity whereas intuitively it should have reduced the productivity. Another weakness in CBT Analyst (Kearsley 1985) is that the effect of an individual productivity adjustment factor is not predictable because of the stepped ranges of threshold values used to select the appropriate *effort to learner time ratio*. For example, the existence of a storyboard only has an effect at the borderline of two ranges of threshold values; changing several productivity adjustment factors produced no apparent effect on the result but one further change doubled the *effort to learner time ratio*. The threshold values are presented because it was impossible to accurately predict the productivity adjustment factor's effect on the *effort to learner time ratio*.

Miles (1990)

The "quick and dirty" courseware effort estimation method (Miles 1990) used six productivity adjustment factors[6] presented as a single grouping. Each of the productivity adjustment factors had an associated description, measurement value and scale type. The values presented for "quick and dirty" productivity adjustment factors are dimensionless. They represent the raw units which are converted to *effort to learner time ratio* at the end of the calculation. This effort estimation method is predictable so it would be a simple task to convert these values into $Dh\ \overline{Lh^{-1}}$ units.

Milette and Trevor-Deutsch (1995)

This study was concerned with comparative *cost* benefit analysis of traditional training against courseware. However, the courseware effort estimation method used eleven productivity adjustment factors[7] presented as a single grouping. The productivity adjustment factors had an associated description, measurement value and a booklet

which provided a more detailed description of the rating scale. The raw values, which are dimension-less, associated with the productivity adjustment factors are presented in this analysis. However, the method is predictable and it would be possible to estimate the effect of individual productivity adjustment factors on *effort to learner time ratios.*

Tan and Nguyen (1993)

Tan and Nguyen (1993) were concerned with capital and recurrent *cost* benefit analysis of courseware development. Their courseware effort estimation method used only two productivity adjustment factors[8]. A book described the method and associated productivity adjustment factors in more detail. The values of the productivity adjustment factors are measured in $Dh\ \overline{Lh^{-1}}$ units.

Mikos et al. (1987b)

In their study of expert estimation of courseware development effort Mikos et al. (1987b) provided respondents with a list of nine ungrouped productivity adjustment factors[9] to amend their original estimates. Descriptions of these productivity adjustment factors were provided with initial example values; however, these were not published. The descriptions provided were not fully developed in the published studies.

Golas (1993)

Golas' (1993) courseware effort estimation method used fifteen productivity adjustment factors[10]. The productivity adjustment factors were presented under the following two groupings.

- Baseline (2)
- Variables (13)

These productivity adjustment factors have associated descriptions, measurement values and scale types which are measured in $Dh\ \overline{Lh^{-1}}$ units.

Bergman and Moore (1990)

The "Q" factor courseware effort estimation method proposed by Bergman and Moore (1990) used only three productivity adjustment factors[11]. They were described as *"Q" factor* values whose effect was to tailor the multimedia effort to the development environment and complexity of the product. The method for choosing a particular *"Q"*

factor value was not well defined. As indicated previously, the values associated with *"Q" factors* are measured in $\$\overline{Lh^{-1}}$ units.

Marshall et al. (1994)

The multimedia effort estimation method made use of twenty four productivity adjustment factors[12] arranged into the following four groupings.

- Course difficulty (3)
- Interactivity (14)
- Development environment (5)
- Subject expertise (2)

The productivity adjustment factors values were rated using a dimension-less five point scale. The resulting values was converted into $Dh\ \overline{Lh^{-1}}$ units through the use of a dimensionality constant.

4.1.3 Analysis of the productivity adjustment factors

There was considerable overlap in the terms used to describe the productivity adjustment factors which made it difficult to draw meaningful conclusions. The first task in the analysis of the productivity adjustment factors was to classify them into a number of related groupings.

Existing productivity adjustment factor groupings

Six groupings used to classify productivity adjustment factors were found in the literature (Gery 1987; Jay et al. 1987; Schooley 1988; Senbetta 1991; Golas 1993; Marshall et al. 1994). These schemes normally tried to classify productivity adjustment factors based on the author's opinion of the most appropriate grouping. There was no reason given to support the inclusion or exclusion of a productivity adjustment factor in any particular grouping. Analysis of these productivity adjustment factor groupings found that there were problems and inconsistencies. For example, the same or very similar productivity adjustment factors were placed in separate groups by the same author (Gery 1987; Jay et al. 1987). There was also a tendency to combine productivity adjustment factors such as those relating to the client and the developer under one grouping (Gery 1987; Senbetta 1991).

Classification process

Rather than use one of the existing groupings a bottom up approach to classification was adopted. Existing models and classification schemes were analysed to identify the range of potential productivity adjustment factors. The productivity adjustment factors were then grouped together in relation to the attribute of the courseware development that they described. A summarised description of the productivity adjustment factor along with the attribute being measured and associated scale, where known, were constructed from the original descriptions. An iterative approach was taken to check and re-classify the productivity adjustment factor and associated attributes and scale. The original eight broad attribute groupings were reduced to six and the original five hundred and fifty-four productivity adjustment factors reduced to seventy-seven. The resulting groupings and individual productivity adjustment factors were then re-checked against the original descriptions for omission or corruption during the classification process.

The following six broad productivity adjustment factor groupings evolved during the classification process.

- Course
- Subject matter
- Quality
- Technical
- Client
- Developer

In the following Sections these groupings will be used to discuss the productivity adjustment factors found in the twelve studies reviewed.

4.2 COURSE PRODUCTIVITY ADJUSTMENT FACTORS

The course grouping brings together productivity adjustment factors associated with the overall design and interactivity of the courseware to be produced. It includes productivity adjustment factor headings such as the total project size, presentational type, course specification, instructional method, interactivity, testing, and screen design features which determine the overall nature of the courseware to be produced. Table 4.2 shows the seven broad course productivity adjustment factor headings together with the thirteen descriptions of individual productivity adjustment factors found in the twelve studies reviewed. For each productivity adjustment factor the percentage use by the twelve studies is shown in the final column.

Table 4.2 - Reported use of course productivity adjustment factors

Course PAF headings	Descriptions of individual productivity adjustment factors	Reported by study											
		A	B	C	D	E	F	G	H	I	J	K	L	%
Total project size	Total project size (learner hours)	✓	✓					✓	✓		✓			42
Presentation level	Presentation level (rating)						✓				✓			17
Course specification	Analysis status (rating)					✓				✓	✓			25
	Prototype courseware status (rating)										✓			8
Instructional method	Courseware type (rating)	✓	✓	✓	✓	✓		✓		✓		✓		67
	Courseware complexity (rating)	✓						✓				✓		25
	Courseware breakdown (% of each type)	✓			✓									17
Interactivity	Learner control (rating)	✓	✓	✓	✓	✓								42
	Conditional branching (rating)	✓	✓	✓	✓	✓							✓	50
Testing	Question type (rating)	✓		✓									✓	25
	Response analysis (rating)		✓	✓		✓		✓						33
	Feedback (rating)	✓		✓		✓		✓					✓	42
Screen	Design (rating)		✓	✓									✓	25
Use of the thirteen course PAF by individual study (%)		62	46	54	31	46	8	38	8	15	31	15	31	32

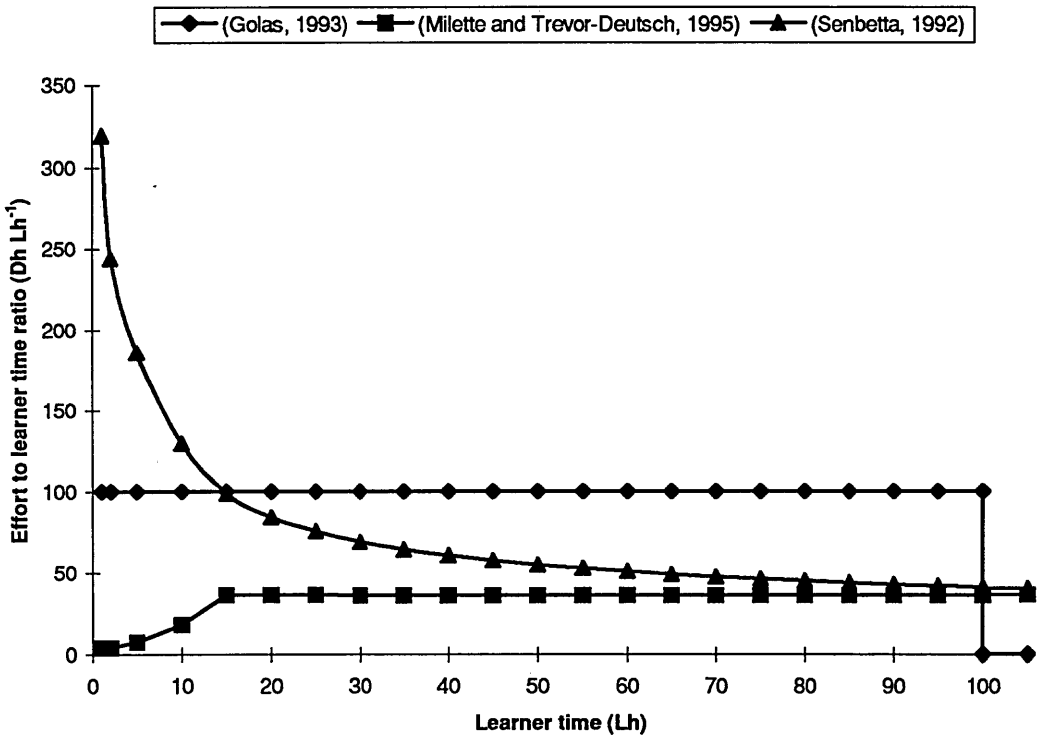
4.2.1 Total project size

Software engineering effort estimation models normally applied the economies of scale coefficient directly as a power of the size factor. In contrast, existing courseware effort estimation methods related economies of scale indirectly to the size factor by using an individual productivity adjustment factor.

Five of the studies reviewed recommended the use of total project size measured in *learner hours* as a productivity adjustment factor (Jay et al. 1987; Senbetta 1991; Golas 1993; Tan and Nguyen 1993; Milette and Trevor-Deutsch 1995). Unfortunately, the effect on development *effort to learner time ratio* of longer projects was inconclusive.

Milette and Trevor-Deutsch (1995) implied that smaller projects were more productive than longer projects. However, this contradicts both Golas (1993) and Senbetta (1992) who proposed that it was longer projects which were more productive. Figure 4.1 shows the effect of total project size on *effort to learner time ratios* for three of the effort estimation methods studied.

Figure 4.1 - Effect of total project size on *effort to leaner time ratios* used by three studies



As Figure 4.1 shows, the reported effects of total project size on *effort to learner time ratios* are dramatically different. However, it should be remembered that these effort estimation methods are primarily based on expert opinion and they are not based on empirical evidence. The difference between the graphs can be partially explained by looking at the development context. Golas' (1993) method is intended to be used in the context of large armed forces projects where anything less than 100 *Lh* is classed as small. On the other hand, both Milette and Trevor-Deutsch (1995) and Senbetta (1992) come from commercial backgrounds in which projects greater than 10 *Lh* would be classified as large. Similar contradictory effects of total project size on effort have been found in software metrics research (Banker and Kemerer 1989; Banker et al. 1994).

Some studies indicate that the effect of a large project is to increase the productivity in comparison to that generated by a number of smaller independent projects of equivalent size. However, the opposite effect has also been found in a number of studies (Kitchenham 1992).

There is little direct empirical evidence to support economies or diseconomies of scale with regard to the total size of a courseware development project. The only evidence comes indirectly from studies into the use of templates in large projects. Blalock (1994) described the work of a team who created development templates and shells to support future courseware production at American Airlines.

> "The set of templates, development shells and models resulting from those months of effort were implemented in Learning Centre's [courseware] project to create 150 hours of interactive courseware. The shell and models did perform as designed. They enabled the developers to create Type III multimedia courseware in record time - fully recouping the initial nine month investment and bringing significant economies of scale to bear on subsequent developments." (Blalock 1994)

The 150 *Lh* courseware project had a productivity of 0.008 $\overline{Lh}\,Dh^{-1}$ *in* comparison to the 0.004 $\overline{Lh}\,Dh^{-1}$ achieved in traditionally-developed projects. In large projects effort expended developing templates and shells can result in lower effort expenditure in individual modules of the total course. However, once the templates and shells are developed the beneficial effects apply to all future projects no matter what their size. The beneficial effect of creating templates and establishing standards can be seen even in relatively small projects. Avner (1988) found that there was a 30% to 43% improvement in productivity between the first and second courseware module depending on the size of the team. He stated that most of this increase could be attributed to "... the result of effort to identify or create tools to be used in subsequent modules".

Arguments can be put forward to explain both reduction and increase in *effort to learner time ratios*. In a large project the opportunity exists for the development team to

prepare house styles, templates and other tools to support the development of later modules. This is especially true if all the modules have a similar format. *Effort* spent on these activities early on in the project will normally be regained in later modules by improved *productivity* and reduced effort. The alternative effect of a large project could be to increase the development effort if a number of independent teams are used in the development. Communication overheads are likely to grow as the number of teams involved increases and the result will be a reduction in productivity and increase in the effort required to complete the project.

The effect on productivity of total project size is difficult to separate from other productivity adjustment factors such as team experience, tool experience or standards. Further experimental research is needed to clarify the situation experimentally.

4.2.2 Presentation level

The presentation level productivity adjustment factor provides a shorthand method of describing the courseware to be developed. It combines a number of individual productivity adjustment factors which some of the other studies described separately. The use of the presentation level productivity adjustment factor in courseware effort estimation methods stems from its use in USA military and government courseware Request for Technical Proposals (RTP) and other documentation (US Navy 1987; Department of the Airforce 1993). Although these standards are old, they still reasonably clearly define the type and level of courseware to be developed.

Only two of the studies found in the literature used presentational level as a productivity adjustment factor (Golas 1993; Miles and Griffith 1993). Both used three levels to describe the interactivity, delivery method and types of media experienced by the learner as they used the courseware as shown in Table 4.3. Detailed descriptions of the presentation levels can be found in Table 4.4 and Table 4.5.

Table 4.3 - Presentation level productivity adjustment factor used by two studies

Study	Description	Scale	Values
(Golas 1993)	Presentation level	I Basic presentation II Medium simulation presentation III High simulation presentation	30,75,200 75,125,250 200,400,600
(Miles and Griffith 1993)	[Courseware] level	1 Baseline presentation 2 Medium simulation presentation 3 High simulation presentation	0 1 3

As Table 4.3 shows, both methods increased the *effort to learner time ratio* as the presentation level increased. Golas (1993) used one of three different values for each presentation level depending on the domain of the courseware's behavioural objectives. Miles and Griffith (1993), on the other hand, used the levels directly without associating them with the behavioural objectives domain. There was no empirical evidence presented by either study to support the choice of values. However, Golas' (1993) range is based on the results of an exercise to collect the opinion of twenty courseware development experts. Each of these experts were asked to check the values used against their own courseware development data. Miles and Griffith (1993) indicated that values used in their effort estimation method had been validated against internal project data.

Table 4.4 - Presentation level I description used by two studies

Level	Description (Department of the Airforce 1993)	Description (US Navy 1987)
I Base	This is the lowest level of interactive courseware development in that Level I lessons are linear (one idea after another), and are used primarily for introducing an idea or concept. There is little "interaction" other than the student touching the screen or using a keystroke or mouse click to continue. The media used are primarily text and graphics (not complex).	General: • Information/knowledge type lessons • Simple questioning techniques • Low [learner] interactivity (basically page turning) • No multi-tasking required • Rudimentary remediation • No real time simulation • No mathematically-driven modelling required Video and minor text presentation: • Limited branching • Real time events presented with location type video using the videodisk media • Minor graphics/text overlay on video (ie headings, captions) Graphics and minor text presentation (No video): • Computer-generated graphics (CGG) presentation • Predominantly simple text and simple CGG pictures • Restricted to simple geometric animation, text • Use of magnetic [disk](s), floppy disk(s) or videodisk media

Table 4.5 - Presentation level II and III descriptions used by two studies

Level	Description (Department of the Airforce 1993)	Description (US Navy 1987)
II Medium simulation	This presentation level involves recall of more information than basic Level II presentation and allows the student to have increased control over lesson presentation; i.e. there is more interaction, such as using a light pen to rotate a switch. CMI is used in Level II lessons to track and analyse student performance. Level II normally combines audio, video, text, graphics and animation.	• Combined information and skill lessons • Moderate degree of programming • Training interactivity with various I/O devices • Computer Managed Instruction (CMI) to track and analyse student performance • Normally combines video and graphics presentation
III High simulation	This level involves aspects of both Level I and Level II while using the full abilities of interactive courseware. Level III may present on-screen interaction similar to that used in an aircraft simulator. This level provides a high degree of interactivity, extensive branching capability, maximum remediation opportunity (supports multiple levels of errors), real-time event simulation with minor equipment limitations, capability to interface with other output devices, and [extensive] CMI capability.	• Primarily used for procedural task/skills • High student interactivity • Extensive branching capability (falls short of artificial intelligence) • Maximum remediation opportunity (i.e. multiple responses measure degree of error and give relevant [feedback]) • Real time simulation with minor equipment limitations (i.e. timing sequences or start-up switch changes) • Capability to interface with other output devices • Exhaustive CMI capability

4.2.3 Course specification

The course specification productivity analysis factor heading looked at how well defined was the courseware to be developed (Kearsley 1985; Mikos et al. 1987b; Golas 1993). The underlying assumption was that the existence of Training Needs Analysis (TNA) documentation or prototype courseware reduced the *effort to learner time ratio* required to create the courseware. Without clearly defined aims and objectives effort must be expended clarifying the situation with the client during the actual project. An incomplete or poorly defined courseware specification may hamper the development. However while this makes sense, no empirical evidence was presented to support this assumption.

In terms of the measurement paradigm the assumption is that courseware specification is excluded from the measurement baseline. This is not the same as a full analysis but is enough to specify what is being done. Full Training Needs Analysis and development of prototypes would normally occur in the instructional design phase. In this case the *effort to learner time ratio* should be reduced if the Training Needs Analysis documentation and courseware prototypes already exist.

Analysis status

Kearsley(1985) described the status of a "storyboard" using a four point scale the existence of which reduced the threshold value used to estimate the *effort to learner time ratio*. The other three values were rated as zero and so did not affect the threshold value. This is similar to the binary decision used by Golas (1993) where inadequate TNA documentation increased the *effort to learner time ratio*. Table 4.6 shows the analysis status productivity adjustment factors used by two of the effort estimation methods reviewed.

Table 4.6 - Analysis status productivity adjustment factors used by two studies

Study	Description	Scale	Value
(Kearsley 1985)	Storyboard	Exists Partial None Unknown	-5 0 0 0
(Golas 1993)	Training needs analysis documentation	Inadequate Adequate	20 0

As Table 4.6 shows, the net effect for both studies was to increase the *effort to learner time ratio* if there was inadequate Training Needs Analysis documentation. This is in agreement with Mikos et al. (1987b) who used "content definition level" as one of the nine factors in their Project Complexity Multiplier (PCM).

Prototype courseware status

Golas(1993) used a productivity adjustment factor which looked at the existence of prototype courseware. The development of a prototype would indicate that some preparatory analysis had been carried out and design decisions taken. It is also likely that outline templates for elements of the courseware could be produced from the prototype. As Table 4.7 shows, no prototype increased the *effort to learner time ratio* by 50 $Dh\ \overline{Lh^{-1}}$.

Table 4.7 - Prototype courseware status productivity adjustment factors used by one study

Study	Description	Scale	Value
(Golas 1993)	Prototype courseware	Does not exist Exists	50 0

As Table 4.7 shows, this is a flat rate increase that applies no matter what the size of the project. However, it is likely that the beneficial effects of a prototype disappear after the

first few modules in a long project where earlier modules effectively serve as prototypes for later modules.

4.2.4 Instructional method

The course instructional method productivity adjustment factor heading brings together three closely related measures and scale descriptions found in the studies reviewed. All the existing courseware effort estimation methods related the choice of instructional method used to development effort. No empirical evidence was presented to support this assumption but it is reasonable to assume that entering questions into a certification test generator requires less effort than producing a high fidelity simulation of the same size.

Courseware type

Productivity adjustment factors related to courseware type were used by eight of the studies reviewed (Kearsley 1985; Gery 1987; Jay et al. 1987; Mikos et al. 1987b; Schooley 1988; Bergman and Moore 1990; Senbetta 1992; Milette and Trevor-Deutsch 1995). Table 4.8 shows productivity adjustment factors used by the five effort estimation methods.

Table 4.8 - Courseware type productivity adjustment factors used by five studies

Study	Description	Scale	Value
(Gery 1987)	Instructional design strategy	Drill and practice, linear tutorial Branched tutorial, linear case study Simulation, highly branched tutorials	85-150 150-300 300+
(Milette and Trevor-Deutsch 1995)	Lesson style	Page turner Tutorial Simulation	1 5 10
(Schooley 1988)	Functional mode	Certification test Drill and practice Tutorial Simulation	10-150 30-400 100-1000 250-1500
(Kearsley 1985)	Type of courseware	Don't know Testing Tutorial Embedded Simulation	0 +1 +1 +5 +5
(Bergman and Moore 1990)	Type of multimedia	Training Kiosk presentation	0.1 2.5

As Table 4.8 shows, four of the methods used an overlapping set of scales which range from testing to simulation. All these methods increased the *effort to learner time ratio*

as they moved from testing through to simulation. Bergman and Moore (1990) used a different scale which reflects the wider range of multimedia applications they normally developed.

Courseware complexity

Courseware complexity was used by three of the studies reviewed (Jay et al. 1987; Bergman and Moore 1990; Milette and Trevor-Deutsch 1995). In all these cases it appears to cover very similar ground to the courseware type previously described. However, it allows the courseware type to be rated in terms of its complexity. For example, a page turner could be described as unsophisticated, or having basic features, or complex (Milette and Trevor-Deutsch 1995). The unsophisticated version may contain only linear sequences while the sophisticated page turner could be used to describe a hypertext environment. There is an intuitive relationship between courseware complexity productivity adjustment factors and effort but no empirical evidence was provided to support the assumption. Table 4.9 shows the complexity productivity adjustment factors found in the effort estimation methods.

Table 4.9 - Complexity productivity adjustment factors used by two studies

Study	Description	Scale	Value
(Bergman and Moore 1990)	Custom training	Simple Complex	0.1-5 1.5-5
(Milette and Trevor-Deutsch 1995)	Software sophistication	Unsophisticated Basic features Complex	1-3 4-7 8-10

Courseware type breakdown

Courseware type breakdown was used in two of the studies reviewed (Jay et al. 1987; Schooley 1988). In both cases it was intended to allow different courseware types to be used in the delivery of the finished course. The other studies assumed that only one courseware delivery method was used or that the most complex type was considered for estimation purposes (Kearsley 1985). Table 4.9 shows the courseware type breakdown productivity adjustment factor used by Schooley (1988).

Table 4.10 - Courseware type breakdown productivity adjustment factors used by one study

Study	Description	Scale	Value
(Schooley 1988)	Functional mode breakdown	Certification test	0-100%
		Drill and practice	0-100%
		Tutorial	0-100%
		Simulation	0-100%

4.2.5 Interactivity

The interactivity heading brings together learner control and conditional branching productivity adjustment factors found in the studies reviewed. Both of these productivity adjustment factors were assumed to be related to the development effort required to produce the courseware. The underlying assumption was that it required more effort to produce courseware which had greater learner control than courseware which had very little. Similarly, the type of conditional branching built into courseware is assumed to influence development effort. No experimental evidence was presented to support these assumptions but they are reasonable conclusions.

Learner control

Learner control is suggested by five of the studies reviewed (Kearsley 1985; Gery 1987; Jay et al. 1987; Schooley 1988; Senbetta 1991). Schooley (1988) made use of instructional sophistication in his effort estimation method. This brought together three productivity adjustment factors which other authors described separately. The logical design aspect described the type of learner control to be used in the proposed courseware. As Table 4.11 shows, there was a tendency to relate the type of learner control directly to the value used to calculate the *effort to learner time ratio*. The underlying assumption was that the more control the learner has over the courseware during delivery, the greater the effort required to develop the material.

Table 4.11 - Learner control productivity adjustment factors used by three studies

Study	Description	Scale	Value
(Gery 1987)	Interactivity	Low Medium High	85-150 150-300 300+
(Kearsley 1985)	Learner control	Don't know Low degree Moderate degree High degree	0 0 0 +5
(Schooley 1988)	Instructional sophistication - logical design	Didactic Discovery Fading Socratic Dialogue	0.1 0.2 0.4 1

Conditional branching

Conditional branching was recommended by six of the studies reviewed (Kearsley 1985: Schooley , 1988 #196; Gery 1987; Jay et al. 1987; Senbetta 1991; Marshall et al. 1994). Schooley's (1988) physical design aspect is presented here because it was related to the type of conditional branching used in the courseware. All the factors described in Table 4.12 effectively increased the effort expended as the complexity of the conditional branching increased. The assumption was that courseware with little or no branching required minimal effort by the developer while complex branching or the introduction of adaptive branching or student modelling techniques increased the effort required. Again, this makes intuitive sense but is not wholly supported by the empirical results.

Orey, Trent, Young and Sanders (1994) described the use of off-the-shelf hypermedia packages to develop Intelligent Tutoring System (ITS) courseware. They found that, using an appropriate authoring tool, it took 155 *Dh* to create the prototype simulation and Intelligent Tutoring System. Student interactions lasted 3 *Lh* which resulted in an effective productivity of 0.0194 $\overline{Lh}\,Dh^{-1}$. Even allowing for the fact that this was a prototype, the result is better than the productivity achieved creating linear testing courseware. Marshall et al. (1994) used the results of a linear testing courseware which had an effective productivity of 0.0040 $\overline{Lh}\,Dh^{-1}$. The effect of conditional branching is probably dependent on several other productivity adjustment factors such as choice of authoring and productivity tools. Without further research it is difficult to isolate the effect of conditional branching from numerous other productivity adjustment factors.

Table 4.12 - Conditional branching productivity adjustment factors used by four studies

Study	Description	Scale	Value
(Gery 1987)	Conditional branching	Low - proscribed learning paths Medium High - highly individualised and conditionally branched program	85-150 150-300 300+
(Kearsley 1985)	Branching	Don't know Simple linear Moderately complex Very complex	0 0 0 +5
(Schooley 1988)	Instructional sophistication - physical design	Linear Branching Regenerative Multi-track Adaptive	0.1 0.2 0.4 1 4
(Marshall et al. 1994)	Level of interactivity	Linear Simple branching Complex branching Simple adaptive Complex adaptive	1 2 3 4 5

4.2.6 Testing

The testing heading brings together three related productivity adjustment factors found in the studies reviewed. In each case the underlying assumption was that the effort required to develop courseware is related to the complexity of the questioning analysis and feedback to the learner. These three individual productivity adjustment factors are highly interrelated and it is doubtful if they can be treated in isolation. Their effect may also depend on other productivity adjustment factors such as content, authoring and productivity tools and developer experience.

Question type

Only three of the studies reviewed used question type as a productivity adjustment factor (Gery 1987; Jay et al. 1987; Marshall et al. 1994). Question type looked at the level of assessment or testing required in the finished courseware. As Table 4.13 shows, the question type productivity adjustment factors proposed by two effort estimation methods increased the development effort as the complexity of the questioning style increased.

Table 4.13 - Question type productivity adjustment factors used by two studies

Study	Description	Scale	Value
(Gery 1987)	Testing requirements	Low - simple testing Medium High - complex testing or learner assessment	85-150 150-300 300+
(Marshall et al. 1994)	Majority of question style	True/false Multiple choice Single words Limited free text Other	1 2 3 4 5

The Marshall et al. (1994) scale indicated that capturing the learner response and analysing free text responses to questions was more difficult than presenting the same test in a multiple choice question format. While this may be true, there are other questioning styles which would be placed in the "Other" category which require very little effort using the correct authoring tool. Touch and click questions which consume large amounts of effort on one authoring tool require no more effort than multiple choice questions on another. The question type productivity adjustment factor cannot be viewed in isolation. The choice of authoring tool and the existence of templates determine the difficulty experienced creating different types of tests.

Response analysis

Response analysis rates the expected depth to which the questions used in the courseware were analysed. Only four of the studies reviewed made reference to productivity adjustment factors related to response analysis (Kearsley 1985; Gery 1987; Senbetta 1991; Milette and Trevor-Deutsch 1995). Table 4.14 shows the response analysis productivity factors used by (Kearsley 1985; Gery 1987).

Table 4.14 - Response analysis productivity adjustment factors used by two studies

Study	Description	Scale	Value
(Gery 1987)	Response analysis complexity	Low - basic low level response analysis Medium High - complex response analysis	85-150 150-300 300+
(Kearsley 1985)	Answer analysis	Don't know Simple Complex	0 0 +5

Neither of the scales are particularly well defined; but their general effect was to reduce productivity as the complexity of the response analysis increased. This was not wholly supported by the available evidence (Orey et al. 1994).

Feedback

Feedback describes the extent to which the questions and other activities in the courseware provide helpful advice on performance to the learner. Questions which have no feedback are assumed to require less development effort than questions which provide detailed advice and remediation. Five of the studies reviewed used productivity adjustment factors related to feedback (Kearsley 1985; Gery 1987; Jay et al. 1987; Marshall et al. 1994; Milette and Trevor-Deutsch 1995). Table 4.15 shows the feedback productivity adjustment factors used by existing courseware effort estimation methods.

Table 4.15 - Feedback productivity adjustment factors used by four studies.

Study	Description	Scale	Value
(Gery 1987)	Nature and depth of feedback	Low - limited feedback Medium High	85-150 150-300 300+
(Kearsley 1985)	Response feedback	Don't know Simple Mixed Complex	0 0 0 +5
(Milette and Trevor-Deutsch 1995)	Degree of remediation	Questions only Questions with right/wrong answer Pass/fail test at end Pass/fail test at end and repeat course Questions with remediation	1 2 or 3 4 5 6 to 10
(Marshall et al. 1994)	Type of question feedback	None Right/wrong Right/wrong with right feedback Right/wrong with relevant feedback Right/wrong with remediation and feedback on each wrong answer	1 2 3 4 5

4.2.7 Screen design

The screen design heading looked at the overall complexity and creativity required to produce the courseware. The assumption is that courseware which has creative or complex interface design required more effort to develop than courseware which used a well-established and simple house-style. Only three of the studies reviewed made any reference to this aspect of the courseware (Gery 1987; Senbetta 1991; Marshall et al. 1994).

Gery (1987) and Marshall et al. (1994) both used productivity adjustment factors relating to screen design. Like many of Gery's (1987) scales, creativity is incomplete; only the high value has an associated example which is not very useful. Table 4.16.

shows the productivity adjustment factors used by two courseware effort estimation methods.

Table 4.16 - Screen design productivity adjustment factors used by two studies

Study	Description	Scale	Value
(Gery 1987)	Creativity	Low Medium High - highly creative	85-150 150-300 300+
(Marshall et al. 1994)	Complexity of interface	Simple text-based Complex text-based Simple graphics-based Complex graphics-based Windowing graphics-based	1 2 3 4 5

4.3 SUBJECT MATTER PRODUCTIVITY ADJUSTMENT FACTORS

The subject matter grouping brings together productivity adjustment factors associated with the content of the courseware to be produced. It includes productivity adjustment factor headings such as behavioural objectives, content, and learner characteristics, all of which influence the nature of the material to be produced. Table 4.17 shows the three broad productivity adjustment factor headings together with the eleven individual productivity adjustment factors found in the twelve studies reviewed.

Table 4.17 - Reported use of subject matter productivity adjustment factors

Subject matter PAF headings	Descriptions of individual productivity adjustment factors	Reported by study											
		A	B	C	D	E	F	G	H	I	J	K	L	%
Behavioural objectives	Level (rating)	✓	✓	✓									✓	33
	Domain (rating)										✓			8
	Number (rating)												✓	8
Content	Existing material (rating)	✓	✓	✓	✓	✓	✓	✓					✓	67
	Complexity (rating)	✓	✓	✓		✓	✓				✓			50
	Stability (rating)	✓				✓					✓			25
Learner characteristics	Homogeneity (rating)	✓	✓											17
	Expectations (rating)			✓										8
	Computer literacy (rating)	✓												8
	Prerequisites (% of learners)	✓												8
	Developer knowledge (rating)										✓			8
Use of the eleven subject PAF by individual study (%)		64	36	36	9	27	18	9	0	0	36	0	27	22

4.3.1 Behavioural objectives

The behavioural objectives productivity adjustment factor heading brings together three related measures found in the studies reviewed. This is in line with the measurement paradigm and, as part of the courseware specification phase, the main behavioural objectives should be prepared for the courseware to at least module level. Five of the studies reviewed suggested that the level or domain of the behavioural objectives to be taught had an effect on courseware development effort (Gery 1986; Jay et al. 1987; Senbetta 1992; Marshall et al. 1994).

Level

Four of the studies reviewed suggested relating the level of objectives to the effort required to develop the courseware (Gery 1986; Jay et al. 1987; Senbetta 1992; Marshall et al. 1994). Table 4.18 shows the behavioural objectives level productivity adjustment factors used by two effort estimation methods (Gery 1986; Marshall et al.

1994).

Table 4.18 - Behavioural objectives level productivity adjustment factors used in two studies

Study	Description	Scale	Value
(Gery 1987)	Learning objectives	Low - recognition/recall/knowledge Medium High - synthesis, analysis, extrapolation	85-150 150-300 300+
(Marshall et al. 1994)	Level of course objectives	Concrete concepts Abstract concepts Lower order principles Higher order principles Problem solving	1 2 3 4 5

Gery's (1986) objectives in Table 4.18 were based on Bloom's taxonomy of behavioural objectives whilst Marshall et al. (1994) followed Gagne and Briggs' (1979) levels for behavioural objectives. The difference is not critical and is primarily a matter of preference and experience. Both methods assumed that lower level objectives required less developer effort while higher level objectives required more. Unfortunately, no empirical evidence was provided to support this assumption.

Domain

Only Golas (1993) looked beyond the cognitive domain to estimate the effort associated with courseware designed to produce psychomotor skills and attitudinal behavioural changes. A rating scale relates the domain to the basic *effort to learner time ratio* to be used for knowledge, skill and attitudinal courseware developments. There was an underlying assumption that courseware which presented only cognitive objectives required less development effort than courseware which presented skill or attitudinal objectives. However, while it is intuitively correct, no empirical evidence was provided to support this assumption. There appeared to be no simple relationship between the *effort to learner time ratios* used for the three domains and the three levels of presentation.

Table 4.19 - Domain productivity adjustment factors used in one study

Study	Description	Scale	Value
(Golas 1993)	Type of training	Knowledge Skill Attitude	30,75,200 75,125,400 200,250,600

Number

The number of behavioural objectives has already appeared in Chapter 4 as a potential

sizing factor for courseware. However as Table 4.20 shows, Marshall et al. (1994) in MEEM used productivity adjustment factors based on the number of behavioural objectives in the courseware.

Table 4.20 - Behavioural objectives number productivity adjustment factor used in one study

Study	Description	Scale	Value
(Marshall et al. 1994)	Number of course objectives	Less than 20	1
		21 to 40	2
		41 to 60	3
		61 to 80	4
		Greater than 80	5

4.3.2 Content

A number of authors have recommended the use of a content productivity adjustment factor in courseware effort estimation. The assumption was made that creating courseware whose subject matter content already existed and had been presented by an instructor would require less development effort than a new course whose content had to be structured and developed from scratch. Similarly, it was assumed to be more difficult to produce courseware if the subject matter was changing or was in the process of being developed rather than for a course whose content was stable.

Existing material

Eight of the studies reviewed recommended the use of productivity adjustment factors related to the existence of learning materials (Kearsley 1985; Gery 1987; Jay et al. 1987; Schooley 1988; Miles 1990; Senbetta 1991; Marshall et al. 1994; Milette and Trevor-Deutsch 1995). Table 4.21 describes the six existing material rating scales used in courseware effort estimation methods. All these methods increased the development effort required if the courseware was being developed from scratch with no material previously in existence. This was normally the worst case situation and the existence of well-structured materials will result in higher productivity.

Table 4.21 - Existing material productivity adjustment factors found in six studies

Study	Description	Scale	Value
(Gery 1987)	Existing materials	Low - structured comprehensive training materials available Medium High - only unstructured training materials [available]	85-150 150-300 300+
(Schooley 1988)	Pedagogical base	Existing materials available No existing materials available	True False
(Kearsley 1985)	Existing course	Existing course New course Don't know	0 +5 0
(Miles and Griffith 1993)	Existing materials	Yes Some No	-1 0 0
(Milette and Trevor-Deutsch 1995)	Development	Good materials already exist Some materials exist but [they] will need [to be adapted] New materials	1 2-9 10
(Marshall et al. 1994)	Existing course materials	Rewrite of existing multimedia materials Rewrite of existing [courseware] materials Rewrite of written materials Rewrite of tutor-delivered materials New course	1 2 3 4 5

Complexity

Subject matter complexity was recommended by six of the studies as a productivity adjustment factor (Kearsley 1985; Gery 1987; Jay et al. 1987; Miles 1990; Senbetta 1991; Golas 1993). However as Table 4.22 shows, only four studies provided descriptions of the productivity adjustment factors used.

Table 4.22 - Complexity productivity adjustment factors used by four of the studies

Study	Description	Scale	Value
(Gery 1987)	Nature and complexity	Low - simple materials Medium High - complex, conceptual	85-150 150-300 300+
(Kearsley 1985)	Complexity of learning task	Complex learning task Simple learning task Task complexity unknown	+2 0 0
(Miles 1990)	Task complexity	Simple Average Complex Highly complex	-1 0 2 4
(Golas 1993)	Subject matter	Highly complex Not highly complex	100 0

Unfortunately, subject matter complexity is difficult to rate because it is dependent on the person doing the rating; courseware with circuit theory as its subject matter may be "highly complex" to most people but to an electronic engineer it would be considered

"simple".

Stability

Only three of the studies suggested that the stability of the subject matter was a potential productivity adjustment factor (Kearsley 1985; Jay et al. 1987; Golas 1993) although Jay, Bernstein and Gunderson (1987) did make reference to it twice. Table 4.23 shows the subject matter stability productivity adjustment factors recommended by Kearsley (1985) and Golas (1993).

Table 4.23 - Stability productivity adjustment factors from two studies

Study	Description	Scale	Value
(Kearsley 1985)	Subject matter availability	Content available (only added if it is a new course)	+5
		Content is being developed	0
		Content status unknown	0
(Golas 1993)	Instructional content	Stable	0
		Unstable	100

4.3.3 Learner characteristics

Knowledge and experience of the learners for whom the courseware is to be developed is seen by many developers as an advantage in developing courseware. The learner characteristics heading brings together five individual productivity adjustment factors found in the literature.

Homogeneity

Jay, Bernstein and Gunderson (1987) recommended that the homogeneity of the group of learners in relation to the subject matter of the course should be used as a productivity adjustment factor. This is included on the basis that it requires less effort to prepare a course for learners with the same basic knowledge of the subject than it does for a disparate collection of learners. If the learners are at a similar standard and have similar aims then you need only develop the core courseware. On the other hand, learners with different backgrounds and motivation may require additional remediation and extra effort expended to make the course interesting to them. Senbetta's (1991) respondents rate "diversity of target audience" as fifth out of twelve "client factors" which impacted on the effort required to develop courseware. Unfortunately, none of the existing courseware estimation methods make use of homogeneity.

Expectations

Gery (1987) made use of productivity adjustment factors which rated the level of learner expectations. In common with most of Gery's (1987) productivity adjustment factors they are not clearly defined. However, the expectations productivity adjustment factor suggests that it will require more effort to produce courseware when the learners have high expectations both about using the courseware and about learning the subject matter; learners who have low expectations about the courseware may be willing to accept what they get. The result may be to require a lower level of effort for learners with lower expectations of the courseware and subject matter. Table 4.24 shows the learner expectations productivity adjustment factor proposed by Gery (1987).

Table 4.24 - Expectations productivity adjustment factor used by one study

Study	Description	Scale	Value
(Gery 1987)	Learner expectations	Low - low learner expectations Medium High - high learner expectations	85-150 150-300 300+

Computer literacy

Jay, Bernstein and Gunderson (1987) stated that the computer literacy of the learners undertaking the course could have an effect on the development effort requirement. However, no scale or indication of the effect could be found in the literature.

Prerequisites

Jay, Bernstein and Gunderson (1987) suggested that the possession by the learners of prerequisite knowledge, skill and attitudes had an effect on the development effort.

Developer knowledge

If a developer has developed courseware for the target learners in the past then the developers are likely to be more productive than those who have no experience of the learners. Golas (1993) used the developer knowledge productivity adjustment factor in her effort estimation method as shown in Table 4.25.

Table 4.25 - Developer knowledge productivity adjustment factor used by one study

Study	Description	Scale	Value
(Golas 1993)	Target audience	Developer not familiar with target audience Developer familiar with target audience	10 0

4.4 QUALITY PRODUCTIVITY ADJUSTMENT FACTORS

The quality grouping brings together productivity adjustment factors associated with quality assurance aspects of the courseware to be produced. It includes the productivity adjustment factors for quality assurance, quality indicators, standards, and final product features which all influence the overall quality of the courseware to be produced. Table 4.26 shows the four productivity adjustment factors together with the thirteen individual productivity adjustment factors found in the twelve studies reviewed.

Table 4.26 - Reported use of quality productivity adjustment factors

Quality PAF headings	Descriptions of individual productivity adjustment factors	Reported by study											
		A	B	C	D	E	F	G	H	I	J	K	L	%
Quality assurance	Process reliability (rating)	✓	✓											17
	Delay introduced (elapsed time)		✓											8
	Review-revision cycles (number)		✓											8
	Reviewers (number)	✓	✓											17
Quality indicators	Bug level (% of total screens)	✓												8
	Meaningful interactions (% of total screens)	✓												8
	Element of good instruction	✓												8
	Completion rate (%)	✓												8
	Mastery score gain (%)	✓												8
Standards	Standards (rating)	✓		✓										17
Final product	Intended use (rating)					✓		✓			✓			25
	Revisions (% per year)					✓								8
	Deadline date (rating)	✓	✓							✓				25
Use of the thirteen quality PAF by individual study (%)		69	38	8	0	15	0	8	0	0	8	0	0	12

4.4.1 Quality assurance

The quality assurance heading brings together four individual productivity adjustment factors found in the literature related to quality assurance. None of the studies indicated the effect on productivity or provided descriptions and scales. The obvious assumption would be that a reliable quality assurance process would reduce the maintenance effort at the expense of increasing the development effort. However, there is no evidence to support this assumption and a reliable quality assurance process may actually reduce development effort by identifying and eliminating faults earlier on in the life-cycle. Unfortunately, no empirical evidence was provided to indicate the effect of quality assurance on the development effort.

Process reliability

Two studies recommended using the reliability of the quality assurance process as a

productivity adjustment factor (Jay et al. 1987; Senbetta 1991). Jay, Bernstein and Gunderson (1987) mentioned various aspects related to quality reliability three times in their list of productivity adjustment factors but did not describe them in detail. The two references by Senbetta (1991) are described in the next paragraph. As indicated earlier, the overall effect of the reliability of the quality assurance process is unclear. The process itself undoubtedly increases the effort expended during the development; however, if the process is reliable it will detect specification, design and content faults earlier on in the life-cycle. This could reduce the need to rectify problems identified in the testing phase which may subsequently take more effort to deal with than if they had been identified earlier. It seems that an unreliable quality assurance process just adds effort without the beneficial effects.

Delays introduced

Senbetta's (1991) respondents rated "delays in team quality assurance reviews and dependability of results" as seventh out of seven "resource factors" in his Delphic study. His respondents also rated "delays in client's quality assurance reviews" tenth out of twelve "client factors" which affect development effort. Delays in themselves in a well-managed project should not increase the effort although they may increase the elapsed time. This is especially true when the development team are working on a number of simultaneous projects or modules. Any delay introduced in one project is used by an effective manager to concentrate on other modules or projects. This is perhaps the reason why respondents to Senbetta's (1991) study did not rate this productivity adjustment factor highly in terms of their effect on development effort.

Review-revision cycles

Senbetta's (1991) respondents rated the "number of team review-revision cycles" fifth out of seven "resource factors" which affected courseware development effort However, his respondents also rated "number of client review-revision cycles" as second out of twelve "client factors" which affected courseware development effort. The relatively low rating of the internal or "team" review-revision cycle is in general agreement with the reliability of the quality assurance factor rating. Beyond a certain number the more reviews that are introduced into a development, the more effort is consumed in the review process. This may not be balanced by savings in wasted effort

later in the life-cycle correcting faults which could have been removed earlier. The higher rating by the respondent of external or "client" reviews may be due to the possibility that the more opportunity given to a client to review the courseware, the more likely they are to request changes.

Reviewers

Jay, Bernstein and Gunderson (1987) indicated that the number of reviewers involved in the quality assurance process can affect the development effort. This is in line with Senbetta's (1991) finding that his respondents rated the "multiple decision makers" fourth out of twelve "client factors" which affected courseware development effort. It appears that the more reviewers or decision makers involved in the quality assurance process, the more difficult it is to reach a consensus. This can result in changes being introduced to the courseware which would not have occurred had only one person made the decision. From personal experience of trying to develop material for a consortium of twelve independent companies the effort expended redoing work agreed at previous reviews was considerable. In this particular project, the membership of the review committee changed at each meeting as did the internal reviewers based in each company. This made it almost impossible to reach decisions that would remain in force for any length of time. In projects where only one person was involved in the review and decision making process little effort was wasted once the client's requirements and preferred method of working had been established.

4.4.2 Quality indicators

The need to produce courseware which achieves predefined measures of quality may require more effort than courseware which has no such stringent requirements. Within the context of the proposed measurement paradigm there are predefined levels of product and educational quality to be achieved by the courseware. Only Jay, Bernstein and Gunderson (1987) indicated that quality indicators could affect the development effort required to produce multimedia courseware. Unfortunately, they did not provide examples of the rating scale or an indication of the effect on effort for the following five quality indicator productivity adjustment factors:

- Bug level
- Meaningful interactions
- Elements of good instruction
- Completion rate
- Mastery score gain

The mastery score gain and completion rates are defined in the measurement paradigm and increased effort would be expended only when these values were higher than the standard values. The elements of good instruction productivity adjustment factor appears to be based on Gagne and Briggs' (1979) events of instruction. Unfortunately, Jay, Bernstein and Gunderson (1987) did not describe this or any of the other productivity adjustment factors in any detail. It is not clear what their effect would be or what were the normal or expected values.

4.4.3 Standards

Only two authors indicated that the existence of quality standards could have an effect on the development effort (Gery 1987; Jay et al. 1987). Jay, Bernstein and Gunderson (1987) indicated that quality standards and specifications were important but did not provide an example of their effect on the development effort. Table 4.27 shows the standards productivity adjustment factor proposed by Gery (1987).

Table 4.27 - Standard productivity adjustment factor used in one study

Study	Description	Scale	Value
(Gery 1987)	Courseware specification standards	Low Medium High	85-150 150-300 300+

As Table 4.2 shows, the scales were not fully developed by Gery (1987) and do not entirely make sense. However, intuitively the low rating should apply to fully developed standards while the high rating should be applied when there are no standards.

4.4.4 Final product

The final product heading brings together three individual productivity adjustment factors found in the literature. The underlying assumption was that the quality of the final product had an effect on the effort expended developing the courseware. However,

no empirical evidence was provided to support this assumption.

Intended use

The intended use of the finished courseware would be expected to have an effect on the development effort required to construct multimedia courseware. Logically, developing courseware as a working prototype would require less effort than developing a commercial product. Similarly, the need to develop courseware to a higher standard than that assumed by the measurement paradigm would increase the amount of effort required. However as Table 4.28 shows, only three authors suggested that intended use should be used as a productivity adjustment factor (Kearsley 1985; Golas 1993; Milette and Trevor-Deutsch 1995).

Table 4.28 - Intended use productivity adjustment factors used by three studies

Study	Description	Scale	Value
Kearsley, 1985 #135	Intended use	Internal use only Commercial product Both Don't know	0 +5 +5 0
(Milette and Trevor-Deutsch 1995)	Finished product	Internal use only Commercial quality	1-5 6-10
(Golas 1993)	MIL-STD specification	Best commercial quality acceptable MIL-STD specification required	0 50

The effect in all three cases was to increase the development effort for higher specification courseware. This would have to be adjusted to fit the measurement paradigm where commercial quality is the norm.

Revisions

Kearsley (1985) suggested that the intended number of revisions per year would have an effect on the development effort. As Table 4.29 shows, the higher the expected percentage annual revision, the greater the effort expended in its creation. Again, this makes sense in that courseware with a high annual percentage revision has to be designed and developed in such a way as to make it easy to revise at a later date. Effort expended doing this in the initial development will reduce the effort required later making changes.

Table 4.29 - Revision productivity adjustment factor used by one study

Study	Description	Scale	Value
(Kearsley 1985)	Revision	Under 5% annually	+1
		5-20% annually	+3
		Over 20% annually	+5
		Don't know	0

Deadline

The adequacy of the delivery schedule or urgency of deadline were suggested as a potential productivity adjustment factor by three of the authors of studies found in the literature (Jay et al. 1987; Mikos et al. 1987b; Senbetta 1991). Neither Jay, Bernstein and Gunderson (1987) nor Mikos et al. (1987b), described the exact effect of deadline on effort. The respondents in Senbetta's (1991) Delphic study rated "client's schedule (time and deadline)" as eighth out of twelve "client factors" which had an effect on development effort. This low rating indicates that the critical effect of deadline was not universally shared by all respondents in the study.

The effect of deadline has support from software engineering effort estimation models. Boehm (1981) included required development schedule (SCED) as a productivity adjustment factor in his software engineering-based COCOMO effort estimation model. In this example both too tight or over-generous deadlines increase the effort required to complete the project.

There is some empirical evidence to support the inclusion of deadline as a productivity adjustment factor. Avner (1984) observed one hundred and forty three independent courseware development teams and found that deadline date was the "... best single predictor of production rate and the most reliable method of controlling production rate." He also claimed that the effect of a known completion deadline was so great as to invalidate any data collected to evaluate the predictive accuracy of an effort estimation method (Avner 1994). The example given by Avner (1984) concerned the adoption of a new authoring tool which increased the productivity of one team with the result that they were able to produce courseware modules in a fraction of the time previously taken. No improvements in productivity were registered and courseware was delivered to an original deadline set prior to the adoption of the new authoring tool; the team members just used the liberated effort to catch-up on other projects which were behind schedule. This is very similar to the effect described by Boehm (1981).

4.5 TECHNICAL PRODUCTIVITY ADJUSTMENT FACTORS

The technical grouping brings together productivity adjustment factors associated with the media and software and hardware aspects of the courseware to be produced. It includes authoring tools, productivity tools, delivery systems, media, and courseware features which all influence the technological environment in which the courseware will be produced. Table 4.30 shows the five productivity adjustment factor headings together with the twelve descriptions of individual productivity adjustment factors found in the twelve studies reviewed.

Table 4.30 - Reported use of technical productivity adjustment factors

Technical PAF headings	Description of individual productivity adjustment factors	Reported by study											
		A	B	C	D	E	F	G	H	I	J	K	L	%
Authoring tools	Type (rating)	✓	✓	✓		✓	✓						✓	50
	Stability (rating)	✓									✓			17
	Productivity class (rating)				✓									8
Productivity tools	Productivity tools (rating)	✓		✓	✓	✓								33
Delivery systems	Peripherals (rating)	✓	✓	✓				✓	✓					42
	Resources (rating)	✓									✓			17
	Capabilities (rating)	✓	✓	✓										25
	Price (rating)	✓		✓										17
Media	Video and audio (rating)	✓			✓	✓						✓	✓	42
	Graphics, animation and simulation (rating)	✓		✓		✓		✓					✓	42
Courseware features	Record keeping (rating)	✓	✓	✓										25
	Help facilities (rating)	✓	✓											17
Use of the twelve technical PAF by individual study (%)		92	42	58	25	33	8	17	8	0	17	8	25	28

4.5.1 Authoring tools

The authoring tools heading brings together four related productivity adjustment factors found in the studies reviewed. It was assumed that the choice of authoring tool had an effect on the courseware development effort. However, this assumption was not wholly supported in the literature. Avner (1984) found no difference in productivity in the long term related to the choice of courseware authoring tool because experienced developers produced templates and tools to overcome any limitations or problems associated with any one particular tool. The choice of authoring tools therefore only has an effect in the early stage of its use. It seems likely that, as courseware developers gain experience with a tool, they recognise its limitations, develop ways around the problem or find other tools which do the job better. However despite this, a number of productivity adjustment factors were suggested which showed the relationship between the choice of

authoring tools and productivity.

Type

Six of the studies reviewed concluded that the type of authoring tool selected had an effect on productivity and hence on development effort (Kearsley 1985; Gery 1987; Jay et al. 1987; Miles 1990; Senbetta 1991; Marshall et al. 1994). Table 4.31 shows the four authoring tools type productivity adjustment factors used by Miles (1990), Gery (1987), Kearsley (1985) and (Marshall et al. 1994).

Table 4.31 - Type productivity adjustment factors used by four studies

Study	Description	Scale	Value
(Gery 1987)	Authoring tools	Low - menu-driven tools	85-150
		Medium - authoring systems interface to a programming language	150-300
		High - authoring/programming language	300+
(Kearsley 1985)	Courseware developed	Using a programming language	+3
		Using an authoring language	+3
		Using an authoring system	+1
		Don't know	0
(Miles 1990)	Development system	Sophisticated author system	-1
		Authoring system	0
		Authoring language	2
		Programming language	4
(Marshall et al. 1994)	Production environment	Authoring environment	1
		Authoring system	2
		Authoring language	3
		High level language	4
		Low level language	5

While it is almost certainly true that the choice of authoring tool type is important in newly-formed teams, over time its effect is likely to be reduced. Well-established teams can compensate for short comings in the tool by developing templates or add-on tools which overcome the problems. However, all the existing effort estimation methods assume that higher level tools increase productivity which in turn reduces development effort. This is perhaps too simplistic a view in that the suitability of the authoring tool to the task is also important. Writing a simulation with an authoring system designed to produce tutorials is unlikely to be any more productive than writing it in a high level language.

Stability

Both Golas (1993) and Jay, Bernstein and Gunderson (1987) indicated that the stability

of the authoring tool was an important productivity adjustment factor. An authoring tool which is being developed may require re-writes of previously developed courseware to take advantage of new features. Conversion of courseware to a newer version of an authoring tool also requires re-testing to ensure that it still works correctly. As Table 4.32 shows, only Golas (1993) used stability as a productivity adjustment factor in her effort estimation method.

Table 4.32 - Stability productivity adjustment factor used in one study

Study	Description	Scale	Value
(Golas 1993)	ICW software	Beta version of authoring system Stable authoring system	80 0

Productivity class

Only Schooley (1988) in the studies reviewed attempted to provide a productivity adjustment factor which evaluated the productivity class of the authoring tool. In this particular case, the productivity of the tool was rated for each different type of courseware to be developed. So, for example, an authoring tool which was rated as being in the top 20% for tutorials could be in the bottom 20% for simulation elements of the course. The rating was based on the developer's opinion of the tool's capabilities and its effect on productivity. However as Table 4.33 shows, selecting different classes produced remarkably large differences in the ranges of values used in effort calculations.

Table 4.33 - Productivity class productivity adjustment factors used in one study

Study	Description	Scale	Value
(Schooley 1988)	Productivity tool class	A Class (Top 20% of tools) B Class C Class D Class E Class (Bottom 20% of tools)	10-500 11-750 12-1000 13-1250 15-1500

4.5.2 Productivity tools

As well as authoring tools, most courseware developers have a range of additional tools, templates and libraries which they use in the production of courseware. The productivity tools heading brings together individual productivity adjustment factors related to the existence of templates, libraries and other tools which enhance the productivity of a development team (Kearsley 1985; Gery 1987; Jay et al. 1987; Schooley 1988).

Jay, Bernstein and Gunderson (1987) and Gery (1987) treated each element in isolation and rated graphics, multimedia interfaces, productivity tools and development tools individually. Schooley, (1988) on the other hand, used mechanical design to group together all four aspects. It is quite difficult to separate these different approaches so Table 4.34 presents the productivity adjustment factors used in the three courseware effort estimation methods.

Table 4.34 - Productivity tools productivity adjustment factors used by three studies

Study	Description	Scale	Value
(Gery 1987)	Multimedia interfaces	Low - integrated into authoring system Medium - external multimedia interfacing tool High - no multimedia interfacing tools	85-150 150-300 300+
(Gery 1987)	Productivity tools	Low - powerful editors, automated design tools Medium High - no editors or productivity tools	85-150 150-300 300+
(Gery 1987)	Development tools	Low - structured development tools Medium High - no development tools	85-150 150-300 300+
(Gery 1987)	Graphics library	Low - graphics library available Medium High - custom graphics	85-150 150-300 300+
(Schooley 1988)	Library saving fraction (%)	Utility library Application library Graphics library Character set	0-100 0-100 0-100 0-100
(Kearsley 1985)	Courseware and graphics library	Library exists No library Library unknown	-5 0 0

As Table 4.34 shows, the productivity tools productivity adjustment factor heading brings together a disparate range of different individual productivity adjustment factors. It includes the availability of media interfacing and instructional design tools as well as courseware templates and media libraries. The existence of these at the start of a project may improve productivity and reduce the development effort. However, it is only likely to improve productivity if the tools are relevant to the courseware being developed (Jay et al. 1987).

4.5.3 Delivery system

The delivery system productivity adjustment factor heading brings together four related aspects of the hardware and software package used to create and deliver the courseware.

Peripherals

In the early days of multimedia the integration of external devices to provide video or other capabilities could consume a large proportion of the courseware development effort. Non-standard interfaces with obscure control languages made it difficult to reliably control an external device. While the situation is certainly easier today, the existence of any non-standard peripheral can still cause problems. Five of the studies reviewed referred to peripherals as productivity adjustment factors (Gery 1987; Jay et al. 1987; Senbetta 1991; Tan and Nguyen 1993; Milette and Trevor-Deutsch 1995). As Table 4.35 shows, only three of the studies actually provided rating scales.

Table 4.35 - Peripherals productivity adjustment factors used by three studies

Study	Description	Scale	Value
(Gery 1987)	Other media integration	Low Medium High	85-150 150-300 300+
(Milette and Trevor-Deutsch 1995)	Peripherals	None Regular screen and keyboard Regular screen, keyboard and printer External devices, touch screen etc. Specialist (Videodisk)	1 1 or 2 3 4 to 9 10
(Tan and Nguyen 1993)	CBIL category	Level 1 - standard hardware Level 2 - simple external devices Level 3 - complex external devices	18 60 180

Resources

Starting a project without the required resources in place is never a good idea and this is particularly true in the development of multimedia courseware. Two of the studies reviewed proposed the use of productivity adjustment factors related to the availability of resources (Jay et al. 1987; Golas 1993). Jay, Bernstein and Gunderson (1987) suggested the use of two productivity adjustment factors. The first of these tried to determine "non-development tasks to be included in the project", in other words what other activities have to be done that don't contribute directly to the development of the courseware. Writing a peripheral device driver because the hardware manufacture does not provide one for your authoring tool is an example of a non-development task. It has to be done to complete the project and increases the apparent effort without actually producing any courseware product. Jay, Bernstein and Gunderson (1987) also indicated that the pressure that a team were under from management to complete the project without the necessary resources should be used as a productivity adjustment factor.

Golas (1993) used the existence of resources as a productivity adjustment factor in her effort estimation method (see Table 4.36).

Table 4.36 - Resources productivity adjustment factor used in one study

Study	Description	Scale	Value
(Golas 1993)	Resources	In place at start of project Not in place at start of project	0 20

Capabilities

It is easier to develop multimedia courseware for a powerful delivery platform than it is to squeeze the last ounce of performance out of an under-power system. Optimising the performance of courseware so that it will run quickly on an under-powered computer can consume large amounts of development effort. Hence the capability of the delivery system was seen as important by three of the studies reviewed (Gery 1987; Jay et al. 1987; Senbetta 1991). However as Table 4.37 shows, only Gery (1987) produced a productivity adjustment factor related to the capabilities of the delivery system. Unfortunately, it is vague and there are no examples to support the choice of one rating on the scale over another.

Table 4.37 - Delivery system productivity adjustment factor used in one study

Study	Description	Scale	Value
(Gery 1987)	Delivery hardware capability and limitations	Low Medium High	85-150 150-300 300+

Price

Two of the studies indicated that the price of the delivery system had an effect on the productivity and hence on effort (Gery 1987; Jay et al. 1987). Since Jay, Bernstein and Gunderson's (1987) study is an extended list of productivity adjustment factors based on Gery's (1987) original list this is hardly surprising. Unfortunately, neither fully explained the relationship between the price of the delivery system and effort. It can only be assumed that the more expensive the delivery system, the fewer a developer is likely to purchase for development and testing purposes. The result is the courseware may be developed on existing systems and ported to the actual delivery system only when necessary. On a large project this could result in delays being introduced due to various teams queuing to use the delivery system. It may also result in courseware having to be re-written to improve its performance on the actual delivery system. If this

is the case then it does not provide any more information than the resources productivity adjustment factor does. Table 4.39 shows the productivity adjustment factor used in Gery's (1987) effort estimation method.

Table 4.38 - Price productivity adjustment factor used in one study

Study	Description	Scale	Value
(Gery 1987)	Presentation system cost	Low Medium High	85-150 150-300 300+

4.5.4 Media

The greater the range of media used in multimedia courseware, the more effort will be expended in its production and integration. The media productivity adjustment factor heading brings together two individual productivity adjustment factors which were found in the literature.

Video and Audio

Video and audio were suggested as a productivity adjustment factor by five of the studies reviewed (Kearsley 1985; Jay et al. 1987; Schooley 1988; Bergman and Moore 1990; Marshall et al. 1994). Table 4.39 shows the media type productivity adjustment factors used by the four effort estimation methods.

Table 4.39 - Video and audio productivity adjustment factors used by four studies

Study	Description	Scale	Value
(Kearsley 1985)	Interactive media	Interactive audio/video involved Interactive audio/video not involved Interactive audio/video possible	+5 0 0
(Bergman and Moore 1990)	Video	Professional facilities Professional facilities and actors Professional facilities and external locations or celebrity actors	>1 >3 5
(Schooley 1988)	Instructional sophistication - mechanical	Text Graphics Animation Interactive video Database driven	0.1 0.2 2 4 4
(Marshall et al. 1994)	Video requirements	None Existing video Simple original linear video clip Complex original linear video clip Complex original interactive video clip	1 2 3 4 5
(Marshall et al. 1994)	Video density	Less than 1 per 20 frames 1 per 20 frames 1 per 10 frames 1 per frame More than 1 per frame	1 2 3 4 5
(Marshall et al. 1994)	Audio requirements	None Existing audio Simple original audio Complex original audio Extremely complex original audio	1 2 3 4 5
(Marshall et al. 1994)	Audio density	Less than 1 per 20 frames 1 per 20 frames 1 per 10 frames 1 per frame More than 1 per frame	1 2 3 4 5

Graphics, animation and simulation

The type, complexity and volume of graphics, animation and simulation was suggested as a productivity adjustment factor by five of the studies reviewed (Kearsley 1985; Gery 1987; Jay et al. 1987; Marshall et al. 1994; Milette and Trevor-Deutsch 1995). As Table 4.40 shows, only four of the effort estimation methods made use of these factors.

Table 4.40 - Graphics, animation and simulation productivity adjustment factors used by four studies

Study	Description	Scale	Value
(Gery 1987)	Graphics type, complexity and volume	Low - no graphics or line graphics Medium High	85-150 150-300 300+
(Kearsley 1985)	Graphics	Colour/graphics involved No colour/graphics involved Colour/graphics possible	+5 0 0
(Milette and Trevor-Deutsch 1995)	Graphics	Text only Screen captures Simple author drawn Functional Simple animation/simulation Complex animation/simulation	1 2 3 5 5 10
(Milette and Trevor-Deutsch 1995)	Critical path	Graphics [produced] before project Graphics [produced] during project Graphics [produced] at or near the end of the project	1-2 3-7 8-10
(Marshall et al. 1994)	Graphics requirements	None Existing artwork Simple original artwork Complex original artwork Extremely complex original artwork	1 2 3 4 5
(Marshall et al. 1994)	Graphics density	Less than 1 per 20 frames 1 per 20 frames 1 per 10 frames 1 per frame More than 1 per frame	1 2 3 4 5
(Marshall et al. 1994)	Animation requirements	None Existing animation Simple original animation Complex original animation Mathematically accurate animation	1 2 3 4 5
(Marshall et al. 1994)	Animation density	Less than 1 per 20 frames 1 per 20 frames 1 per 10 frames 1 per frame More than 1 per frame	1 2 3 4 5
(Marshall et al. 1994)	Simulation requirements	None Existing simulation Simple original simulation Complex original simulation Realistic simulation	1 2 3 4 5
(Marshall et al. 1994)	Simulation density	Less than 1 per 20 frames 1 per 20 frames 1 per 10 frames 1 per frame More than 1 per frame	1 2 3 4 5

As can be seen from Table 4.40, the range of interpretation about what constitutes a complex graphic is large. Kearsley (1985) found that the result of including any colour or graphics was to increase effort whereas Milette (1995) found that the effort only

increased as the type of graphics involved became more complex and moved into animation or simulation. This demonstrates the effect of time on a productivity adjustment factor. In 1985 a computer that could handle colour graphics was the exception whilst today colour graphics are taken for granted and only increasing the quality and realism of the graphic, or perhaps the accuracy of the animation or simulation, is seen to require increased effort.

4.5.5 Courseware features

The courseware features heading brings together two related individual productivity adjustment factors found in the literature. The premise for these two is that courseware which required more than basic record keeping features and help facilities would require greater effort to produce.

Record keeping

Three of the studies indicated that the record keeping requirements had an effect on the effort required to develop the courseware (Gery 1987; Jay et al. 1987; Senbetta 1991). As Table 4.41 shows, only Gery (1987) provided an example of the scale used to rate its effect.

Table 4.41 - Record keeping productivity adjustment factor used in one study

Study	Description	Scale	Value
(Gery 1987)	CMI requirements	Low Medium High	85-150 150-300 300+

Help facilities

Jay, Bernstein and Gunderson (1987) and Senbetta (1991) both suggested that the more extensive the help facilities provided, the greater the effort required to develop the courseware. Again, neither provided scales to assist in determining the effect of help facilities on productivity.

4.6 CLIENT PRODUCTIVITY ADJUSTMENT FACTORS

The client grouping brings together productivity adjustment factors associated with the client or organisation for which the courseware is to be produced. It includes the productivity adjustment factors for organisation, involvement, and experience which all influence potential delays introduced into the production of the courseware by the client organisation or their representatives. Table 4.42 shows the three productivity adjustment factor headings together with the eight descriptions of individual productivity adjustment factors found in the twelve studies reviewed.

Table 4.42 - Reported use of client productivity adjustment factors

Client PAF heading	Description of individual productivity adjustment factors	Reported by study											
		A	B	C	D	E	F	G	H	I	J	K	L	%
Organisation	Type (rating)	✓	✓							✓				25
	Geographical location (rating)		✓											8
	Staff turnover (%)	✓												8
Involvement	Commitment and availability (rating)	✓	✓	✓										25
	Management skills (rating)	✓								✓				17
	Consistency (rating)	✓	✓								✓			25
	Decision making (rating)	✓	✓			✓								25
Experience	Experience (rating)	✓	✓							✓			✓	33
Use of the eight client PAF by individual study (%)		88	75	13	0	13	0	0	0	38	13	0	13	21

4.6.1 Organisation

Three of the studies reviewed suggested that productivity adjustment factors relating to the client's organisation type, geographical location or employee turnover may affect courseware development effort. This is perhaps too much of a generalisation and there is no empirical evident to support the premise.

Type

Three of the studies made reference to the client's organisation type as a productivity adjustment factor (Jay et al. 1987; Mikos et al. 1987b; Senbetta 1991). While none of these studies provided rating scales, it is relatively clear from the discussion that they are using commercial and non-commercial as the basis for their decision. Senbetta's (1991) respondents rated the "type of client (commercial, educational, government, etc)" as eleventh equal out of twelve "client factors". The effect of different types of organisation was therefore perceived to be very limited. It would appear from the other studies that something other than the profit motive of the client needs to be considered.

Mikos et al. (1987b) described it as "politics/corporate culture" whilst Jay, Bernstein and Gunderson (1987) described it in terms of both organisation type and required customer satisfaction.

Geographical location

Only Senbetta (1991) suggested the use of geographical location as a productivity adjustment factor. In his Delphic study respondents rated "client location (travel requirements)" as eleventh equal from twelve "client factors". This relatively low rating suggests that it is not perceived to have a major effect on development effort.

Employee turnover

This productivity adjustment factor brings together a number of individual references to the expected turnover in client employees such as contact person, decision makers and subject matter experts (Jay et al. 1987). Jay, Bernstein and Gunderson (1987) indicated that the expected turnover in the client's employees had an effect on the effort required to complete the project. However, this would only have an effect if employee turnover was very high during the lifetime of the project. Rapid changes in subject matter experts, decision makers and contact person would make it difficult to maintain consistent decision-making throughout the courseware development. The need to induct the new contact people in to the project would also increase the risk that they may not agree with previous decisions or structures. This could increase the effort expended if previous agreements were changed and the courseware had to be altered.

4.6.2 Involvement

The involvement of the client's contact in the project was viewed as important by four authors (Gery 1987; Jay et al. 1987; Mikos et al. 1987b; Senbetta 1991). A contact person in the client's organisation who was always available and was committed to the success of project would make decisions and resolve problems quickly. This can improve the perceived productivity of the development because delays will not be introduced waiting for clearance from a client's contact who is working on several other tasks or projects. However, on a well-managed project the effect of delays will only increase the elapsed time not effort. A team working on several projects would use any delays productively working on other projects.

Commitment and availability

Productivity adjustment factors related to commitment and availability of the client's contact person were found in three of the studies (Gery 1987; Jay et al. 1987; Senbetta 1991). However as Table 4.43 shows, Gery (1987) was the only one who provided a scale for this productivity adjustment factor. This indicates that the effect of a highly committed client was to reduce the effort required to complete the project.

Table 4.43 - Commitment and availability productivity adjustment factor used in one study

Study	Description	Scale	Value
(Gery 1987)	Client skill commitment and time	Low - high client commitment Medium High - low client commitment	85-150 150-300 300+

Management skills

One of the studies made reference to the client's "personality"; however they did not describe in detail what they meant by the term (Mikos et al. 1987b). It is certainly possible to visualise a situation in which the personality of the client contact's could affect the effort. For example, a client contact's with an aggressive personality could demand changes to the original courseware specification late on in the project. But on the whole personality is probably best absorbed in to a more general productivity adjustment factor called management skills which was proposed by Jay, Bernstein and Gunderson (1987).

Consistency

Client contact's consistency was seen as a potential productivity adjustment factor by three of the studies (Jay et al. 1987; Senbetta 1991; Golas 1993). If clients keep on changing their mind about design features, style or content of the courseware this could increase the effort required to complete the project. Similarly, if clients accept or agree something at one meeting only to want something else at another meeting the effect will be to increase the effort required by having to alter existing work. As Table 4.44 shows, only Golas (1993) described a productivity adjustment factor related to client consistency.

Table 4.44 - Consistency productivity adjustment factor used by one study

Study	Description	Scale	Value
(Golas 1993)	Customer consistency	Is using objective and consistent acceptance criteria	0
		Is not using objective and consistent acceptance criteria	50

Decision making

The ability of the client contact to make decisions about courseware without having to refer to others was seen as important by three of the studies (Kearsley 1985; Jay et al. 1987; Senbetta 1991). Jay, Bernstein and Gunderson (1987) were concerned with the number of people involved in the decision making and review process. From personal experience the more people involved, the more effort is required to get agreement on content, design or structure of courseware. They also suggested looking at the client contact's authority to get decisions taken quickly and to make anyone else involved keep to what they agreed. As shown in Table 4.45, only Kearsley (1985) provided a productivity adjustment factor related to decision making.

Table 4.45 - Decision making productivity adjustment factor used by one study

Study	Description	Scale	Value
(Kearsley 1985)	Approval	Single decision-maker	0
		Decision-maker not clear	+5
		Decision-maker unknown	0

4.6.3 Experience

Four of the studies made reference to the previous experience of the client as a potential productivity adjustment factor (Jay et al. 1987; Mikos et al. 1987b; Senbetta 1991; Marshall et al. 1994). It was assumed that a client who had experience of working with a particular developer would know the development process and less effort would be expended by the developer explaining their methods to the client (Jay et al. 1987; Mikos et al. 1987a; Mikos et al. 1987b). The other underlying assumption was that if a client had previous experience of developing courseware or other flexible learning materials, then there was less need for the developer to expend effort explaining what was happening; having gone through the process once, the client was more likely to know what was happening and to provide information in an appropriate format. However as Table 4.46 shows, only Marshall et al. (1994) used a productivity adjustment factor related to the client's experience of courseware development.

Table 4.46 - Experience productivity adjustment factor used by one study

Study	Description	Scale	Value
(Marshall et al. 1994)	Subject matter expert's multimedia experience	Extensive multimedia development experience	1
		Some multimedia development experience	2
		Extensive [courseware] development experience	3
		Some [courseware] development experience	4
		None	5

4.7 DEVELOPER PRODUCTIVITY ADJUSTMENT FACTORS

The developer grouping brings together productivity adjustment factors associated with the organisation or team developing the courseware. It includes organisation, cost, team, manager, designers, programmers, media experts, and subject matter experts which all influence the capability of the developer to produce the courseware. Table 4.47 shows the eight productivity adjustment factor headings together with the twenty descriptions of individual productivity adjustment factors found in the twelve studies reviewed.

Table 4.47 - Reported use of developer productivity adjustment factors

Developer PAF headings	Description of individual productivity adjustment factors	Reported by study											
		A	B	C	D	E	F	G	H	I	J	K	L	%
Organisation	Type (rating)	✓		✓										17
	Employee turnover (rating)	✓		✓										17
	Experience (rating)	✓						✓					✓	25
	Development procedures (rating)	✓	✓	✓		✓	✓						✓	50
Cost	Overhead rates (%)				✓									8
	Developer effort charge (currency Dh^{-1})				✓			✓				✓		25
	Contribution to project (number)				✓									8
Team	Size (number)			✓	✓									17
	Experience (rating)	✓	✓	✓	✓	✓	✓						✓	58
	Dedication (% of effort on project)	✓		✓										17
	Group dynamics (rating)	✓		✓		✓								25
Manager	Experience (rating)	✓		✓	✓	✓				✓	✓			42
Designers	Experience (rating)	✓			✓	✓				✓	✓			42
Programmers	Experience (rating)	✓	✓		✓	✓		✓			✓			50
	Authoring tool experience (rating)	✓	✓		✓						✓			33
Media experts	Media tool experience (rating)	✓												8
	Courseware experience (rating)	✓												8
Subject matter experts	Availability (rating)	✓	✓					✓		✓	✓		✓	50
	Number (number)	✓												8
	Experience (rating)	✓	✓										✓	8
Use of the twenty developer PAF by individual study (%)		80	30	40	45	30	10	20	0	15	25	5	25	27

4.7.1 Organisation

The organisational aspects of the courseware developer have been used or suggested extensively in the literature as the basis for productivity adjustment factors (Kearsley 1985; Gery 1987; Jay et al. 1987; Miles 1990). They typically looked at the type of organisation and at the infrastructure that existed to support the development of the courseware.

Type

Two of the studies reviewed indicated that the type of organisation in which the development is taking place had an effect on productivity (Gery 1987; Jay et al. 1987). Jay, Bernstein and Gunderson (1987) suggested a total of four different productivity adjustment factors related to the organisation concerned. In addition to the type being classified as either "private, academic or governmental" the following productivity adjustment factors were also suggested:

- Internal politics
- Management pressure
- Internal communications

These appear to be related to a certain extent to the type of organisation. For example, a commercial organisation may have a low internal political and communication rating but high management pressure rating in comparison to another organisation. Gery (1987) proposed two of these productivity ratings as shown in Table 4.48.

Table 4.48 - Type productivity adjustment factors used in one study

Study	Description	Scale	Value
(Gery 1987)	Management pressure	Low Medium High	85-150 150-300 300+
(Gery 1987)	Political factors	Low Medium High - highly political environment	85-150 150-300 300+

Employee turnover

Jay, Bernstein and Gunderson (1987) and Gery (1987) both recommended the use of employee turnover as a productivity adjustment factor. The assumption was that the higher the turnover of employees, the greater the effort that would be expended by new employees learning about the courseware, tools, procedures and standards. This unproductive effort is counted against the overall development project effort. Gery's (1987) employee turnover productivity adjustment factor is shown in Table 4.49. Notice that the scale incorporates the team and client turnover but it indicates that the general effect of high turnovers is an increase in the value of the *effort to learner time ratio.*

Table 4.49 - Employee turnover productivity adjustment factor used in one study

Study	Description	Scale	Value
(Gery 1987)	Team and client turnover	Low - stable client interface Medium High - high team or client turnover	85-150 150-300 300+

Experience

Three of the studies reviewed proposed the use of courseware development experience as a productivity adjustment factor (Jay et al. 1987; Marshall et al. 1994; Milette and Trevor-Deutsch 1995). In Jay, Bernstein and Gunderson's (1987) case, they suggested using "corporate experience in the courseware business" and "successful delivery" as productivity adjustment factors. Table 4.50 shows the scales proposed by two of the effort estimation methods (Marshall et al. 1994; Milette and Trevor-Deutsch 1995).

Table 4.50 - Experience productivity adjustment factors used in two studies

Study	Description	Scale	Value
(Milette and Trevor-Deutsch 1995)	Developer experience	Created more than 3 courseware courses Created 2 or 3 courseware courses First course created	1 5 10
(Marshall et al. 1994)	Development team's multimedia experience	Extensive multimedia development experience Some multimedia development experience Extensive [courseware] development experience Some [courseware] development experience None	1 2 3 4 5

Development procedures

Six of the studies reviewed indicated that the existence of well-defined procedures and standards for developing courseware can improve the productivity in courseware developments (Kearsley 1985; Gery 1987; Jay et al. 1987; Miles 1990; Senbetta 1991; Marshall et al. 1994). If well-defined standards and procedures already exist then they can be used immediately but if they do not exist then they will have to be developed as the project proceeds. This reduces the productivity of the developer who has to expend unproductive effort either redoing existing work once standards become established or actually developing procedures by trial and error.

Avner (1984) found that systematic planning and control procedures were the most commonly observed management techniques among the courseware development teams studied. While these techniques tended to work well within the teams that had

developed them or by the group which had been extensively trained in their use, casual adoption did not produce increased productivity. The use of project management techniques without training or an understanding of their role tended to increase the project management overhead and reduce productivity. Table 4.51 shows the development procedures productivity adjustment factors used in four existing effort estimation methods.

Table 4.51 - Development procedures productivity adjustment factors used by four studies

Study	Description	Scale	Value
(Gery 1987)	Project development methodology	Low - structured project development method	85-150
		Medium	150-300
		High - no structured project development method	300+
(Gery 1987)	Courseware standards	Low - clear and comprehensive	85-150
		Medium	150-300
		High - no courseware standards	300+
(Gery 1987)	Project development methodology	Low - structured project development method	85-150
		Medium	150-300
		High - no structured project development method	300+
(Kearsley 1985)	Guidelines	Courseware guidelines used	0
		Courseware guidelines not used	+5
		Courseware guidelines may be used	0
		Use of guidelines unknown	0
(Miles 1990)	Existing standards	Yes	-1
		No	1
(Marshall et al. 1994)	Instructional design, development and delivery methodology	None	1
		Informal	2
		Formal first generation	3
		Formal second generation	4
		Formal third generation	5

4.7.2 Cost

Cost is not a productivity adjustment factor as such but it was used in three of the effort estimation methods in estimating the *cost* of the courseware to be developed (Schooley 1988; Bergman and Moore 1990; Milette and Trevor-Deutsch 1995). It is presented here for completeness as it is related to the developer and team productivity adjustment factors.

Overhead rates

As shown in Table 4.52, Schooley (1988) used three percentage overhead rates to calculate the developer effort rate charged to the client. These values were set by the

developer to recover overhead costs and effort not directly charged out against the project.

Table 4.52 - Overhead rates used in one study

Study	Scale	Value
(Schooley 1988)	Overhead rate General and administrative rate Fee/Profit Rate	0-200% 0-200% 0-200%

Developer effort charge

Table 4.53 shows the costing factors associated with the three existing effort estimation methods (Schooley 1988; Bergman and Moore 1990; Milette and Trevor-Deutsch 1995). Schooley (1988) was the most advanced allowing individual developer rates to be set for up to four categories of labour rate when calculating the effective developer effort charge from basic wage rates.

Table 4.53 - Developer effort charge rates used by three studies

Study	Scale	Value
(Schooley 1988)	Labour category wage rates Labour category	Currency Dh^{-1} I-IV
(Milette and Trevor-Deutsch 1995)	Developer effort charge	Currency Dh^{-1}
(Bergman and Moore 1990)	Developer effort charge	Currency Dh^{-1}

Contribution to project

This final costing factor worked out the contribution of the individual labour categories to the courseware development (Schooley 1988). Table 4.54 shows the contribution to project costing factors used by Schooley (1988).

Table 4.54 - Contribution to project costing factors used by one study

Study	Description	Scale	Value
(Schooley 1988)	Labour category	Number of Category I Number of Category II Number of Category III Number of Category IV	Number Number Number Number
(Schooley 1988)	Labour category percentage contribution	Percentage contribution of Category I Percentage contribution of Category II Percentage contribution of Category III Percentage contribution of Category IV	% % % %

4.7.3 Team

Most of the studies reviewed used one or more productivity adjustment factors related to the courseware development team. In general, a relationship was proposed between the development experience of the team and the effort required to develop courseware.

Size

Four of the studies reviewed used total team size as a productivity adjustment factor (Gery 1987; Jay et al. 1987; Schooley 1988; Marshall et al. 1994). As Table 4.55 shows, the general consensus is that larger teams are on the whole less productive (Gery 1987; Schooley 1988; Marshall et al. 1994).

Table 4.55 - Size productivity adjustment factors used by two studies

Study	Description	Scale	Value
(Schooley 1988)	Teaming multiplier	Team size (i) Where i is the number in the team	n_1=0.8 n_2=1.0 n_i=i*0.5
(Gery 1987)	Number of development team members	Low - small Medium High - large number of team members	85-150 150-300 300+
(Marshall et al. 1994)	Size of the proposed development team	More than 15 10-15 5-9 2-4 1	1 2 3 4 5

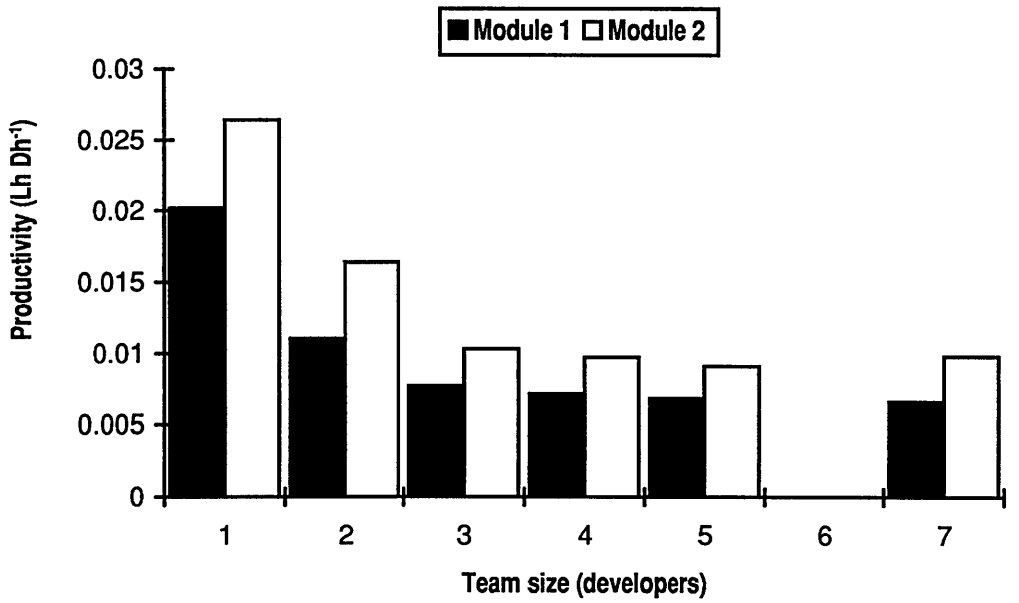
This is supported by empirical evidence. Avner (1988) investigated the effect of team size on the efficiency and quality of courseware. He asked twelve courseware development team leaders to state the optimum team size for production efficiency and quality. Only ten team leaders responded, the other two felt the questions were too simplistic. The results indicated that even experienced team leaders could not agree on the optimum size for a courseware development team. However, their opinions were highly correlated with the size of their own production teams (r=0.917 for efficiency and r=0.889 for quality) and therefore it is difficult to arrive at any valid, unbiased conclusions.

Avner (1988) also reported the results of a study to evaluate the effect of team size on productivity[13]. Using the data from one hundred and forty three courseware development teams, twenty-seven developers who met the following conditions were identified:

- They had at least two years' courseware development experience
- They had produced a minimum of two modules in at least two different sized teams
- They had produced courseware modules in a "parallel team" situation where they had relative autonomy once the specification for the module was set
- They had produced courseware modules as instructional designers

The aim of the criteria was to produce a set of relatively homogeneous participants and to reduce the variability of the results. Figure 4.2 shows the productivity for teams of different sizes for the first and second courseware modules produced.

Figure 4.2 - Productivity against team size (Avner 1988)



Avner (1988) found that the effect of team size on productivity is significant ($F(5,110)=9.77$, $P<0.0005$). Figure 4.2 clearly shows that the effect of the intra-team communications overhead was to reduce productivity as the team size grew from one developer to seven (Garton et al. 1984; Avner 1988).

Experience

The experience heading brings together a number of proposed productivity adjustment factors which were found in the studies reviewed (Kearsley 1985; Gery 1987; Jay et al. 1987; Schooley 1988; Miles 1990; Senbetta 1991; Marshall et al. 1994). All of these

studies indicated that a relationship existed between the experience of the team developing the project and their productivity. The relationship was rated in a number of different ways such as counting the number of previous projects or rating them using some experience scale. However, the effect is always the same; the greater the experience the higher the productivity. Table 4.56 shows the experience productivity adjustment factors used by (Kearsley 1985; Gery 1987; Schooley 1988; Miles 1990; Marshall et al. 1994).

Table 4.56 - Experience productivity adjustment factors used in four studies

Study	Description	Scale	Value
(Gery 1987)	Previous team experience	Low Medium High	85-150 150-300 300+
(Gery 1987)	Team experience	Low Medium High	85-150 150-300 300+
(Schooley 1988)	Experience factor - individual	Experience of developing courseware No experience of developing courseware	True False
(Schooley 1988)	Experience factor - team	Experience of working together No experience of working together	True False
(Kearsley 1985)	Team's previous development experience	Experience of working together No experience of working together Does not apply Team experience unknown	0 +5 0 0
(Miles 1990)	Personnel	Individuals inexperienced New team (<1 year) Old team (>2 year) Separate locations 2 or more above	1 1 -1 1 1
(Marshall et al. 1994)	Development team's multimedia experience	Extensive multimedia development experience Some multimedia development experience Extensive [courseware] experience Some [courseware] experience None	1 2 3 4 5
(Marshall et al. 1994)	Development team's subject matter experience	Expert knowledge of the subject Good knowledge of the subject Some knowledge of the subject Knowledge of related subject No knowledge of subject	1 2 3 4 5

Whatever the actual effect on productivity, the implied relationship between experience and effort is supported by empirical studies. In analysing the productivity of experienced courseware development teams, Avner (1988) noted that there was a significant improvement in productivity between the first and second module of between 30% and 43% depending on the size of the team ($F(1,110)=13.62$, $P<0.0005$).

Avner (1988) stated that most of the increase in productivity between the first and second module could be attributed to "... the result of effort to identify or create tools to be used in subsequent modules". Part of the initial development involves creating tools which can be re-used in the second and subsequent modules. This allows the team to concentrate more productively on the development of courseware.

Jay, Bernstein and Gunderson (1987) also indicated that the percentage of the team requiring training may affect the overall project productivity. This appears to be a variation on the experience productivity adjustment factor and, as such, it should probably be included here rather than on its own. Novice developers are notably less productive than experienced developers but the effect is not long term. Steinberg (1984) reported that after one or two lessons, the effort expended by authors to produce courseware usually decreases and then levels off. Cates (1994) reported that experienced authors could design more rapidly. He claims it is not unusual for experienced developers to "... design simple screens in two-thirds or half of the time of a novice". He also indicated that "... even novice authors, once immersed in the authoring experience, develop facility with the authoring system fairly quickly."

Dedication

The dedication of a team to the project is also assumed to have some effect on productivity (Gery 1987; Jay et al. 1987). As Table 4.57 shows, only Gery (1987) provided a rating scale based on dedication to the project.

Table 4.57 - Dedication productivity adjustment factor used by one study

Study	Description	Scale	Value
(Gery 1987)	Dedication	Low - small % Medium High - large %	85-150 150-300 300+

Group dynamics

The group dynamics productivity adjustment factor brings together three individual factors found in the studies reviewed (Kearsley 1985; Gery 1987; Jay et al. 1987). All of the following appear to be looking at interrelated features of the development team:

- Synergy
- Role clarity
- Motivation

Synergy attempts to rate how well the team will work together; role clarity looks at how clear the individual members were about their individual roles within the team and finally, how motivated were the team to undertake the project. These three factors could all have an effect on the productivity of the project but it is unlikely that they can be treated in isolation. They are therefore presented under a group dynamics productivity adjustment factor heading. Table 4.58 shows the three group dynamic productivity adjustment factors found in the existing effort estimation methods (Kearsley 1985; Gery 1987).

Table 4.58 - Group dynamics productivity adjustment factors used in two studies

Study	Description	Scale	Value
(Gery 1987)	Team synergy	Low Medium High	85-150 150-300 300+
(Gery 1987)	Role clarity	Low - high role clarity Medium High - low role clarity	85-150 150-300 300+
(Kearsley 1985)	Motivation level of team	High motivation level Moderate motivation level Low motivation level Motivation level unknown	-10 0 0 0

4.7.4 Manager experience

Five of the studies reviewed indicated that the experience of the manager affected the productivity of the overall project (Kearsley 1985; Gery 1987; Jay et al. 1987; Schooley 1988; Golas 1993). The reasoning was that an experienced manager would be aware of the problems associated with courseware development and would use this knowledge to improve the productivity of the team. Table 4.59 shows the four existing effort estimation models reviewed which made use of manager experience as a productivity adjustment factor (Kearsley 1985; Gery 1987; Schooley 1988; Golas 1993).

Table 4.59 - Manager experience productivity adjustment factors used in four studies

Study	Description	Scale	Value
(Kearsley 1985)	Management experience	Experienced courseware manager Inexperienced courseware manager Experience of manager unknown	0 +5 0
(Schooley 1988)	Experience factor - Manager	Experienced Inexperienced	True False
(Gery 1987)	Project management skills	Low - strong project management skills Medium High	85-150 150-300 300+
(Golas 1993)	Manager experienced	Experienced Inexperienced	0 100

4.7.5 Designers' experience

The experience of the courseware designers was seen as critical to the successful development of the final course by five of the studies reviewed (Kearsley 1985; Jay et al. 1987; Mikos et al. 1987b; Schooley 1988; Golas 1993). They assumed that experienced designers would be able to save considerable effort by designing courseware which was simple to develop but provided an effective learning experience. Jay, Bernstein and Gunderson (1987) also indicated that the percentage of the development team who had courseware design experience had an effect on productivity. However as Table 4.60 shows, only two of the existing effort estimation methods used the designer productivity adjustment factor (Kearsley 1985; Golas 1993). The effect in both cases of the courseware designer being inexperienced was to increase the effort *to learner time ratio.*

Table 4.60 - Designer experience productivity adjustment factors used in two studies

Study	Description	Scale	Value
(Kearsley 1985)	Designer experience	Courseware experience Some courseware experience No courseware experience	+1 +3 +5
(Golas 1993)	Designer experience	Experienced Inexperienced	0 80

4.7.6 Programmers

The role of the courseware programmers was viewed as critical to the success of the development project. It was generally assumed that the more experienced and competent the programmers are who are involved in the project, the less effort will be expended developing the courseware. However, this was not wholly borne out by the limited experimental evidence available (Garton et al. 1984)[14]. Two aspects of the

programmers' experience were used as productivity adjustment factors (Kearsley 1985; Jay et al. 1987; Schooley 1988; Senbetta 1991; Golas 1993; Milette and Trevor-Deutsch 1995). The first looked at the overall experience of the programmers in developing courseware whilst the second looked at the programmers' experience of using the authoring tool selected for the project.

Programmer experience

Four of the studies reviewed recommended the use of programmer experience as a productivity adjustment factor (Kearsley 1985; Jay et al. 1987; Golas 1993; Milette and Trevor-Deutsch 1995). Table 4.61 shows the programmer experience productivity adjustment factor used by existing effort estimation methods. In each case inexperience increased the *effort to learner time ratio* used in the estimate.

Table 4.61 - Programmer experience productivity adjustment factors use by three studies

Study	Description	Scale	Value
(Kearsley 1985)	Authoring experience	Considerable authoring experience Some authoring experience No authoring experience Authoring experience unknown	-5 0 0 0
(Milette and Trevor-Deutsch 1995)	Programmer experience	Many templates Limited First course	1 5 10
(Golas 1993)	Programmer experience	Inexperienced Experienced	60 0

Authoring tool experience

Authoring language experience with the particular authoring tool to be used was seen as an important productivity adjustment factor by four of the studies reviewed (Jay et al. 1987; Schooley 1988; Senbetta 1991; Golas 1993). Table 4.62 shows the productivity adjustment factors used by Schooley (1988) and Golas (1993).

Table 4.62 - Authoring tool experience productivity adjustment factors used by two studies

Study	Description	Scale	Value
(Schooley 1988)	Experience factor - authoring tools	Experienced Inexperienced	True False
(Golas 1993)	Authoring language	Not familiar Familiar	15 0

4.7.7 Media experts

Only one of the studies reviewed made any reference to the media expertise of the development team (Jay et al. 1987). In this particular case, the media expertise was

described in terms of graphics design which reflects the age of this study. Nevertheless, the effect on development effort of the production of media for multimedia courseware can be large. The following two productivity adjustment factors were identified in the literature.

- Media tools experience
- Courseware experience

Unfortunately Jay, Bernstein and Gunderson (1987) did not fully develop these productivity adjustment factors. The media tools experience appears to rate the media experts' knowledge of the development tools used to create the media whereas the courseware experience was used to rate the media experts' experience of producing courseware.

4.7.8 Subject matter experts

The final productivity adjustment factor found in the literature reviewed related to the availability and experience of subject matter experts in the development team. In defining the measurement paradigm for courseware effort estimation, subject matter experts were treated as part of the development team whether they were client-based or in-house. Five of the studies referred to the subject matter expertise of the development team (Jay et al. 1987; Mikos et al. 1987b; Senbetta 1991; Golas 1993; Marshall et al. 1994). Only Jay, Bernstein and Gunderson (1987) were concerned with the percentage of the development team who were subject matter experts and possessed good writing skills. The other studies were primarily concerned with the number or availability of the subject matter experts (Mikos et al. 1987b; Senbetta 1991; Golas 1993).

Availability

Marshall et al. (1994) reduced the productivity as the availability of the subject matter experts became restricted. Golas (1993) reduced the productivity if the development team had to depend on client-based subject matter experts. This appears to be related to the potentially restricted availability of the subject matter experts. Table 4.63 shows the availability productivity adjustment factors use by these two studies.

Table 4.63 - Availability productivity adjustment factors used by two studies

Study	Description	Scale	Value
(Marshall et al. 1994)	Availability of subject matter expert	Unrestricted contact Daily contact Weekly contact Monthly contact Restricted contact	1 2 3 4 5
(Golas 1993)	SME	In-house Client-based	0 25

Number

Only Jay, Bernstein and Gunderson (1987) suggested that there was a relationship between the number of client-based subject matter experts and development effort. It could be, therefore, that the number of subject matter experts may not be as important as their availability. Having one subject matter expert working full-time on the project may be more beneficial than four people who only worked part-time and were not readily available. Unfortunately, they did not fully discuss the effect of the number of subject matter experts on development effort.

Experience

Marshall et al. (1994) used a productivity adjustment factor which reduced the productivity for subject matter experts who had less experience of developing multimedia or courseware. The use of a subject matter expert experience productivity adjustment factor was endorsed by both Jay, Bernstein and Gunderson (1987) and Senbetta (1991). However as Table 4.64 shows, only one of the courseware effort estimation methods made used of this factor.

Table 4.64 - Experience productivity adjustment factor used in one study

Study	Description	Scale	Value
(Marshall et al. 1994)	Subject matter experts' multimedia experience	Extensive multimedia development experience Some multimedia development experience Extensive [courseware] development experience Some [courseware] development experience None	1 2 3 4 5

4.8 IDENTIFYING THE KEY PRODUCTIVITY ADJUSTMENT FACTORS

The twelve studies related to effort estimation identified seventy-seven individual productivity adjustment factors. However, experience from research into software effort estimation models suggest that a number of these factors are likely to be highly correlated and also using this number of productivity adjustment factors would neither be practical nor desirable in an effort estimation model. Collecting sufficient courseware data to calibrate and validate seventy-seven productivity adjustment factors would also represent an impossible undertaking in all but the largest courseware development organisation. Therefore a subset of key productivity adjustment factors needed to be identified for effort estimation purposes.

4.8.1 Initial ranking

The percentage use of the individual productivity adjustment factor in each of the twelve studies was calculated. This identified the popularity of a productivity adjustment factor and allowed the factors to be ranked from the least popular to the most popular. Productivity adjustment factors with the same percentage use values were awarded equal ranking scores.

The resulting rankings of the productivity adjustment factors are presented in Table 4.65 and Table 4.66. Both tables are ordered in terms of the "All" the "C-K" ranking values which used the results from all twelve studies. The "C-K" columns present the percentage score and ranked order for ten studies excluding Jay, Bernstein and Gunderson (1987) and Senbetta (1991). These two studies contain independent rankings in terms of the individual productivity adjustment factor's effect on effort by one hundred and seventy-nine respondents and twenty-five respondents respectively. Excluding the results from these two studies enabled a comparison to be made between the orders found in this study and these effectively independent studies.

Analysis of the two rankings indicate that the removal of Jay, Bernstein and Gunderson's (1987) and Senbetta's (1991) data had no effect on the top nineteen places in either column. As would be expected, there was most disruption in the mid-table region where the loss of one or two points drastically reduced the percentage use values. Surprisingly, there appears to be very little disruption to the last twenty values in the

table.

Table 4.65 - Top forty productivity adjustment factors ordered by "All" then "C-K" rank column

Group	Headings	Description of individual productivity adjustment factors	All (%)	C-K (%)	All (rank)	C-K (rank)
Subject Matter	Content	Existing material	66.7	60	76.5	76.5
Course	Instructional method	Courseware type	66.7	60	76.5	76.6
Developer	Team	Experience	58.3	50	75	74.5
Developer	Manager	Experience	50	50	71	74.5
Subject Matter	Content	Complexity	50	40	71	68.5
Course	Interactivity	Conditional branching	50	40	71	68.5
Technical	Authoring tools	Type	50	40	71	68.5
Developer	Organisation	Development procedures	50	40	71	68.5
Developer	Subject matter experts	Availability	50	40	71	68.5
Developer	Programmers	Experience	50	40	71	68.5
Technical	Media	Graphics, animation and simulation	41.7	40	64	68.5
Course	Testing	Feedback	41.7	40	64	68.5
Technical	Media	Video and audio	41.7	40	64	68.5
Developer	Designers	Experience	41.7	40	64	68.5
Technical	Delivery system	Peripherals	41.7	30	64	59.5
Course	Interactivity	Learner control	41.7	30	64	59.5
Course	Total project size	Total project size	41.7	30	64	59.5
Course	Testing	Response analysis	33.3	30	58	59.5
Technical	Productivity tools	Productivity tools	33.3	30	58	59.5
Subject Matter	Behavioural objectives	Level	33.3	20	58	50
Developer	Programmers	Authoring tool experience	33.3	20	58	50
Client	Experience	Experience	33.3	20	58	50
Course	Course specification	Analysis status	25	30	47	59.5
Developer	Costing	Developer effort charge	25	30	47	59.5
Quality	Final product	Intended use	25	30	47	59.5
Course	Screen	Design	25	20	47	50
Course	Instructional method	Courseware complexity	25	20	47	50
Course	Testing	Question type	25	20	47	50
Developer	Organisation	Experience	25	20	47	50
Developer	Team	Group dynamics	25	20	47	50
Subject Matter	Content	Stability	25	20	47	50
Developer	Subject matter experts	Experience	25	10	47	31.5
Client	Involvement	Consistency	25	10	47	31.5
Client	Involvement	Decision making	25	10	47	31.5
Client	Involvement	Commitment and availability	25	10	47	31.5
Technical	Delivery system	Capabilities	25	10	47	31.5
Technical	Courseware features	Record keeping	25	10	47	31.5
Quality	Final product	Deadline date	25	10	47	31.5
Client	Organisation	Type	25	10	47	31.5
Course	Presentation level	Presentation level	16.7	20	31	50

Table 4.66 - Bottom thirty-seven productivity adjustment factors ordered by "All" then "C-K" rank column

Group	Headings	Description of individual productivity adjustment factors	All (%)	C-K (%)	All (rank)	C-K (rank)
Developer	Team	Size	16.7	20	31	50
Developer	Organisation	Employee turnover	16.7	10	31	31.5
Client	Involvement	Management skills	16.7	10	31	31.5
Course	Instructional method	Courseware breakdown	16.7	10	31	31.5
Developer	Organisation	Type	16.7	10	31	31.5
Developer	Team	Dedication	16.7	10	31	31.5
Quality	Standards	Standards	16.7	10	31	31.5
Technical	Authoring tools	Stability	16.7	10	31	31.5
Technical	Delivery system	Resources	16.7	10	31	31.5
Technical	Delivery system	Price	16.7	10	31	31.5
Quality	Quality assurance	Reviewers	16.7	0	31	9.5
Subject Matter	Learner characteristics	Homogeneity	16.7	0	31	9.5
Quality	Quality assurance	Process reliability	16.7	0	31	9.5
Technical	Courseware features	Help facilities	16.7	0	31	9.5
Technical	Authoring tools	Productivity class	8.3	10	12	31.5
Course	Course specification	Prototype courseware status	8.3	10	12	31.5
Developer	Cost	Overhead rates	8.3	10	12	31.5
Developer	Cost	Contribution to project	8.3	10	12	31.5
Quality	Final product	Revisions	8.3	10	12	31.5
Subject Matter	Behavioural objectives	Domain	8.3	10	12	31.5
Subject Matter	Behavioural objectives	Number	8.3	10	12	31.5
Subject Matter	Learner characteristics	Expectations	8.3	10	12	31.5
Subject Matter	Learner characteristics	Developer knowledge	8.3	10	12	31.5
Quality	Quality assurance	Review-revision cycles	8.3	0	12	9.5
Client	Organisation	Staff turnover	8.3	0	12	9.5
Quality	Quality assurance	Delay introduced	8.3	0	12	9.5
Client	Organisation	Geographical location	8.3	0	12	9.5
Developer	Media experts	Media tool experience	8.3	0	12	9.5
Developer	Media experts	Courseware experience	8.3	0	12	9.5
Developer	Subject matter experts	Number	8.3	0	12	9.5
Quality	Quality indicators	Bug level	8.3	0	12	9.5
Quality	Quality indicators	Meaningful interactions	8.3	0	12	9.5
Quality	Quality indicators	Element of good instruction	8.3	0	12	9.5
Quality	Quality indicators	Completion rate	8.3	0	12	9.5
Quality	Quality indicators	Mastery score gain	8.3	0	12	9.5
Subject Matter	Learner characteristics	Computer literacy	8.3	0	12	9.5
Subject Matter	Learner characteristics	Prerequisites	8.3	0	12	9.5

4.8.2 Composite rankings

The Jay, Bernstein and Gunderson's (1987) and Senbetta's (1991) rankings of productivity adjustment factors were compared with the "C-K" ranking produced in this study. In addition, a final composite ranking (FCR) was produced by averaging the contribution from the independent three ranking scales. Table 4.67 and

Table 4.68 show the rankings of the productivity adjustment factors from all three sources as well as the final composite ranking (FCR).

Table 4.67 - Top forty productivity adjustment factors ordered by FCR column rank

Group	Headings	Description of individual productivity adjustment factors	C-K (rank)	A (rank)	B (rank)	FCR (rank)
Developer	Team	Experience	74.5	71	76.5	74.00
Subject Matter	Content	Complexity	68.5	76	74	72.83
Course	Interactivity	Conditional branching	68.5	72	72.5	71.00
Technical	Authoring tools	Type	68.5	65	75	69.50
Technical	Media	Graphics, animation and simulation	68.5	73	65.5	69.00
Course	Testing	Feedback	68.5	61	70	66.50
Course	Screen	Design	50	74	72.5	65.50
Course	Testing	Response analysis	59.5	62	71	64.17
Course	Instructional method	Courseware complexity	50	77	63.5	63.50
Developer	Subject matter experts	Availability	68.5	52	58	59.50
Developer	Subject matter experts	Experience	31.5	68.5	76.5	58.83
Client	Involvement	Consistency	31.5	75	68	58.17
Subject Matter	Behavioural objectives	Level	50	60	59	56.33
Technical	Media	Video and audio	68.5	68.5	24	53.67
Developer	Designers	Experience	68.5	66.5	24	53.00
Developer	Organisation	Development procedures	68.5	26	63.5	52.67
Developer	Manager	Experience	74.5	57	24	51.83
Subject Matter	Content	Existing material	76.5	26	53	51.83
Client	Involvement	Decision-making	31.5	55	67	51.17
Technical	Delivery system	Peripherals	59.5	66.5	24	50.00
Client	Involvement	Commitment and availability	31.5	56	60	49.17
Developer	Programmers	Authoring tool experience	50	68	24	47.33
Technical	Productivity tools	Productivity tools	59.5	58	24	47.17
Course	Total project size	Total project size	59.5	53	24	45.50
Course	Interactivity	Learner control	59.5	26	51	45.50
Client	Experience	Experience	50	26	55	43.67
Course	Instructional method	Courseware type	76.5	26	24	42.17
Technical	Delivery system	Capabilities	31.5	26	65.5	41.00
Developer	Organisation	Employee turnover	31.5	63.5	24	39.67
Developer	Programmers	Experience	68.5	26	24	39.50
Client	Involvement	Management skills	31.5	59	24	38.17
Technical	Courseware features	Record keeping	31.5	26	57	38.17
Quality	Final product	Deadline date	31.5	26	54	37.17
Technical	Authoring tools	Productivity class	31.5	54	24	36.50
Course	Course specification	Analysis status	59.5	26	24	36.50
Developer	Cost	Developer effort charge	59.5	26	24	36.50
Quality	Final product	Intended use	59.5	26	24	36.50
Client	Organisation	Type	31.5	26	49.5	35.67
Quality	Quality assurance	Reviewers	9.5	26	69	34.83
Course	Presentation level	Presentation level	50	26	24	33.33

Table 4.68 - Bottom thirty-seven productivity adjustment factors ordered by FCR column rank

Group	Headings	Description of individual productivity adjustment factors	C-K (rank)	A (rank)	B (rank)	FCR (rank)
Course	Testing	Question type	50	26	24	33.33
Developer	Organisation	Experience	50	26	24	33.33
Developer	Team	Group dynamics	50	26	24	33.33
Developer	Team	Size	50	26	24	33.33
Subject Matter	Content	Stability	50	26	24	33.33
Client	Organisation	Staff turnover	9.5	63.5	24	32.33
Quality	Quality assurance	Review-revision cycles	9.5	26	61.5	32.33
Subject Matter	Learner characteristics	Homogeneity	9.5	26	61.5	32.33
Quality	Quality assurance	Process reliability	9.5	26	56	30.50
Quality	Quality assurance	Delay introduced	9.5	26	52	29.17
Client	Organisation	Geographical location	9.5	26	49.5	28.33
Technical	Courseware features	Help facilities	9.5	26	48	27.83
Course	Course specification	Prototype courseware status	31.5	26	24	27.17
Course	Instructional method	Courseware breakdown	31.5	26	24	27.17
Developer	Cost	Overhead rates	31.5	26	24	27.17
Developer	Cost	Contribution to project	31.5	26	24	27.17
Developer	Organisation	Type	31.5	26	24	27.17
Developer	Team	Dedication	31.5	26	24	27.17
Quality	Final product	Revisions	31.5	26	24	27.17
Quality	Standards	Standards	31.5	26	24	27.17
Subject Matter	Behavioural objectives	Domain	31.5	26	24	27.17
Subject Matter	Behavioural objectives	Number	31.5	26	24	27.17
Subject Matter	Learner characteristics	Expectations	31.5	26	24	27.17
Subject Matter	Learner characteristics	Developer knowledge	31.5	26	24	27.17
Technical	Authoring tools	Stability	31.5	26	24	27.17
Technical	Delivery system	Resources	31.5	26	24	27.17
Technical	Delivery system	Price	31.5	26	24	27.17
Developer	Media experts	Media tool experience	9.5	26	24	19.83
Developer	Media experts	Courseware experience	9.5	26	24	19.83
Developer	Subject matter experts	Number	9.5	26	24	19.83
Quality	Quality indicators	Bug level	9.5	26	24	19.83
Quality	Quality indicators	Meaningful interactions	9.5	26	24	19.83
Quality	Quality indicators	Element of good instruction	9.5	26	24	19.83
Quality	Quality indicators	Completion rate	9.5	26	24	19.83
Quality	Quality indicators	Mastery score gain	9.5	26	24	19.83
Subject Matter	Learner characteristics	Computer literacy	9.5	26	24	19.83
Subject Matter	Learner characteristics	Prerequisites	9.5	26	24	19.83

The final composite ranking of the top twenty individual productivity adjustment factors was very stable across the other three study rankings. Although the orderings varied between the rankings, there was reasonable consistency in the selection of the top twenty productivity adjustment factors. Again, in mid-table there are discrepancies in the rankings of a number of factors which appeared in only one or two of the studies. There are also problems with the bottom ten productivity adjustment factors which have

a zero ranking indicating that they were not selected in any of the studies. This was because Jay, Bernstein and Gunderson (1987) only ranked twenty-six productivity adjustment factors while Senbetta (1991) ranked twenty-eight of the seventy-seven productivity adjustment factors identified in this thesis. Comparing the composite ranking factors with the "all" rank shows that there was considerable agreement about the top twenty productivity adjustment factors.

4.8.3 Key productivity adjustment factors

The final composite ranking provided a method of identify potentially important productivity adjustment factors for consideration in effort estimation models. It brings together expert opinion of the factors which affect courseware development productivity based on the opinion of well over two hundred courseware development effort estimation experts from twelve independent studies. However, seventy-seven potential productivity adjustment factors are too many to consider in most courseware development environments. Only those productivity adjustment factors ranked above the third quartile (Q3=50.58) were considered important. Table 4.69 shows the nineteen key productivity adjustment factors which meet this criterion.

Table 4.69 - Nineteen key productivity adjustment factors

Group	Headings	Description of individual productivity adjustment factors	FCR (rank)
Developer	Team	Experience	74.00
Subject Matter	Content	Complexity	72.83
Course	Interactivity	Conditional branching	71.00
Technical	Authoring tools	Type	69.50
Technical	Media	Graphics, animation and simulation	69.00
Course	Testing	Feedback	66.50
Course	Screen	Design	65.50
Course	Testing	Response analysis	64.17
Course	Instructional method	Courseware complexity	63.50
Developer	Subject matter experts	Availability	59.50
Developer	Subject matter experts	Experience	58.83
Client	Involvement	Consistency	58.17
Subject Matter	Behavioural objectives	Level	56.33
Technical	Media	Video and audio	53.67
Developer	Designers	Experience	53.00
Developer	Organisation	Development procedures	52.67
Developer	Manager	Experience	51.83
Subject Matter	Content	Existing material	51.83
Client	Involvement	Decision-making	51.17

As can be seen from Table 4.69, the top nineteen productivity adjustment factors consisted of:

- six from the developer group
- five from the course group
- three from the technical group
- three from the subject matter group
- two from the client group

It is interesting that there are no quality group factors within the nineteen key productivity adjustment factors identified by this technique. It is perhaps surprising that more emphasis was not given to quality by the developers who contributed to the original surveys. Having identified these key factors they will be used in Chapter 6 to investigate their effect on courseware development effort estimation.

4.9 SUMMARY

The productivity adjustment factors found in the twelve studies reviewed on multimedia courseware effort estimation were classified into six groupings of seventy-seven individual factors. In this Section the main findings in this Chapter will be briefly evaluated as will the need for further work in this area.

4.9.1 General effort estimation model

The major finding of this Chapter was that the existing courseware effort estimation models follow the general format for software effort estimation models (Kitchenham 1992; Shepperd et al. 1996). They are described by the equation.

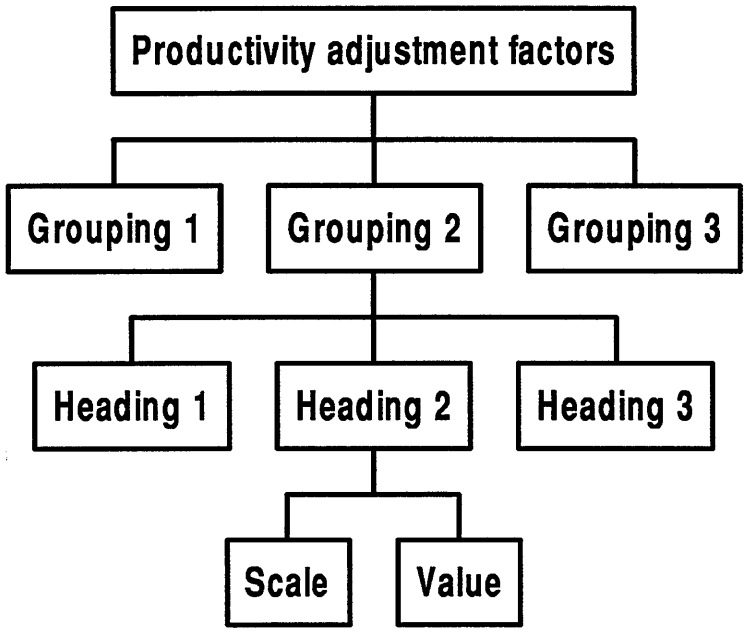
$$Effort = \alpha(Size)^{\beta}$$

Where *size* is an appropriate measurement of courseware size, α is the productivity adjustment factor coefficient and β is an economy of scale exponent. The function of α is to tailor the estimate of productivity to the circumstances which exist in the development environment. In addition, α acts as a dimensional coefficient which converts the units of *size* to the units of effort measured in *Dh*. All the converted forms of the existing effort estimation models, which use *learner time* as their basic size factor, are dimensionally correct and make use of a variety of formats to calculate the productivity adjustment factors which are measured in $Dh\ \overline{Lh^{-1}}$.

4.9.2 Productivity adjustment factor classification

The productivity adjustment factor classification scheme attempts to differentiate between the various factors associated with multimedia courseware development using the hierarchical structure presented in Figure 4.3. Each of these will be reviewed in turn.

Figure 4.3 - Productivity adjustment factors hierarchical structure



Groupings

The six groupings provide a basis for the reclassification of the existing productivity adjustment factors into reasonably exclusive sets. Previous classification schemes tended to combine a number of factors related to the courseware into overlapping groups. This current scheme has the advantage of looking at the course, subject matter, technical and quality factors one by one which enables factors such as the choice of delivery platform to be separated out from factors related to end product quality, content and interactivity. This is an advantage in that it enables course, technical and quality decisions to be separated out from subject matter considerations. The previous schemes tended to mix these elements together making it difficult to re-estimate a development using lower or higher technical, content or quality specifications.

The division of client and developer productivity adjustment factors into two discrete groups also has advantages over existing classification schemes. Previous classification schemes tended to confuse the role played by the client with that played by the developer by combining them under a broad heading such as "human factors". The separation into two groupings is more realistic. Developer factors can be estimated based on a reasonable knowledge of the individual factors which exist within the development environment. The estimates are likely to be more realistic than estimates

about the client which tend to be more speculative. This is especially true if the developer has not worked with the client before.

The most obvious question is whether the proposed groupings are broad enough to cover all aspects of multimedia courseware development. The only obvious missing element is a grouping related to learner or end user of the courseware. In one of the early iterations of the classification process such a group existed, however, it contained so few headings and individual productivity adjustment factors that it was decided to combine it with the quality aspects because they had a number of factors in common. That is not to say that a learner grouping may not be required, just that the productivity adjustment factors from the existing studies did not suggest a need.

Headings

The headings used in the classification process were intended to bring together related sub-groupings of individual productivity adjustment factors. It became apparent during the classification process that a number of individual productivity factors were related and measured various attributes of the same feature. The headings classification also makes clear which aspect of the grouping is being measured by the individual productivity adjustment factor.

Individual

The individual description of the productivity adjustment factor describes the attribute being measured in relationship to the heading. There are a number of common descriptions such as " experience" and "availability" which appear several times and without their associated heading are meaningless. The result of this is that a productivity adjustment factor should always be named in this classification scheme in terms of its grouping, heading and individual description. Also, within the seventy-seven individual descriptions of productivity adjustment factors there are a few factors which appear to cover more than one attribute. The "graphics, animation and simulation" productivity adjustment factor is one example of this. This should probably be split into its individual components but was presented in this format because it was difficult to separate out the scales used in existing effort estimation methods which tended to overlap.

Scales

Every productivity adjustment factor requires a scale by which it can be measured or estimated. Unfortunately, not every study provided scales or examples to aid in the estimation process. Those which did tended to provide scales which required estimation of a factor based on a two to ten point scale. Most of the examples confirmed Jay, Bernstein and Gunderson's (1987) criticism of effort estimation methods which tended to be poorly defined and required the estimator to interpret what was meant by "very experienced", "experienced", "inexperienced" or "very inexperienced". This requirement for interpretation in the existing effort estimation methods is one of the major problems which must be resolved before there is any hope of achieving inter-estimator consistency.

Values

Each example of a productivity adjustment factor which was based on an existing effort estimation method was presented with its associated scale values. To ensure dimensional consistency, these values should be presented as *effort to learner time ratios*. However, some of these existing effort estimation methods did not use these units and it was impossible to estimate the individual effect of a particular value on the final productivity adjustment factor. The result was to make it impossible to compare in detail the effects of individual productivity adjustment factors across different effort estimation models. All that could be stated was that the general outcome of the different values was to increase or decrease productivity. While it was useful to identify similarities and differences between the various outcomes on productivity of the various effort estimation models, it would have been more useful to compare their effects in terms of the *effort to learner time ratio*.

4.9.3 Key productivity adjustment factors

Using only those productivity adjustment factors ranked above the third quartile (Q3=50.58) reduced the original seventy seven factors to a more manageable subset of nineteen. Are the nineteen key productivity adjustment factors shown in Table 4.69 which meet this criterion the best subset to describe factors which can be used to improve accuracy of effort estimation? Probably, but that is not to say that one or more of the productivity adjustment factors lower down in the ranking could not be

important. All that can be said is that the nineteen identified form a reasonable basis for more detailed analysis.

4.9.4 Evaluation

The productivity adjustment factors classification scheme enables the disparate collection of factors found in the twelve studies to be rationalised and reduced to a much smaller set of factors. However, it is not perfect and the requirement to be able to classify all the existing factors under the new scheme placed limitations on the flexibility to rationalise further some of the individual productivity adjustment factors. Despite this limitation, it does form the basis for more work to further rationalise the classification scheme and to introduce less subjective rating scales which describe directly their effect in *effort to learner time* units. This also would form the basis for an experimental evaluation of the effects of individual productivity adjustment factors on courseware development effort. As this study has highlighted, there is very little experimental evidence to support the expert opinion collected in this analysis.

This Chapter has started the process of classifying the productivity adjustment factors found in effort estimation studies and has identified a set of key factors for consideration in the development of courseware effort estimation models. These key productivity adjustment factors will be further rationalised and appropriate scales constructed to enable their effect to be evaluated using courseware development data which will be presented as a case study in Chapter 5.

4.10 REFERENCES

Avner, A., 1988. Is there an ideal size for courseware production teams? In: *30th Association for the Development of Computer-Based Instructional Systems, Philadelphia, 7-10 November*. pp. 143-147.

Avner, A., Smith, S. and Tenczar, P., 1984. CBI authoring tools: effects on productivity and quality. *Journal of Computer-Based Instruction*. **11**: pp. 85-89.

Avner, R.A., 1994. *Personal communication.* (e-mail to I.M. Marshall, 26 July 1994) Urbana-Champaign, IL: University of Illinois, Computer Based Education Research Laboratory.

Banker, R.D., Chang, H. and Kemerer, C.F., 1994. Evidence on economies of scale in software development. *Information and Software Technology*. **36(5)**: pp. 275-282.

Banker, R.D. and Kemerer, C.F., 1989. Scale economies in new software development. *IEEE Transactions on Software Engineering*. **15(10)**: pp. 416-429.

Bergman, R.E. and Moore, T.V., 1990. *Managing interactive video/multimedia projects*. Englewood Cliffs, NJ: Educational Technology Publications.

Blalock, R.H., 1994. How models and templates = Fast and creative development at American Airlines. In: *16 Annual Conference and Exhibition on Interactive Systems for Training, and Job Performance Improvement, Washington Interactive Multimedia 94, Washington, DC,*

Boehm, B.W., 1981. *Software engineering economics*. Englewood Cliffs, NJ: Prentice-Hall.

Cates, W.M., 1994. Estimating the time required to produce computer-based instructional lessons: Descriptive analyses of the production data of novice instructional developers. *Journal of Educational Computing Research*. **10(1)**: pp. 29-40.

Department of the Airforce, 1993. *Information for designers of instructional systems: interactive courseware (ICW) design, development, and management guide.* (AF

Handbook 36-2235, Volume 5), 1 October 1993, Department of the Air Force, Headquarters, US Airforce, Washington DC 20330-5000.

Gagne, R.M. and Briggs, L.J., 1979. *Principles of instructional design.* 2nd Edition ed. New York: Holt, Rinehart & Winston.

Garton, R. et al., 1984. Programming a CBI course: A case study. In: *Proceedings of the Conference of the Association for the Development of Computer-Based Instructional Systems, Columbus, Ohio,* pp. 149-155.

Gery, G., 1986. How long does it really take? Estimating CBT development cost. *DATA Training.* **5(8)**: pp. 31-36.

Gery, G., 1987. *Making CBT happen: Prescriptions for successful implementation of computer-based training in your organisation.* Boston, MA: Weingarten Publications Inc.

Golas, K.C., 1993. Estimating time to develop interactive courseware in the 1990s. In: *13th Interservices Industry Training and Education Conference, Orlando, FL, November 1990.*

Jay, J., Bernstein, K. and Gunderson, S., 1987. *Estimating computer-based training development times.* ARI Technical Report (765), October 1987, Alexandria, VA: Research Institute for Behavioural and Social Sciences.

Kearsley, G., 1985. The CBT advisor: An expert system program for making decisions about CBT. *Performance and Instruction.* **24(9)**: pp. 15-17.

Kitchenham, B.A., 1992. Empirical studies of assumptions that underlie software cost-estimation models. *Information and Software Technology.* **34(4)**: pp. 211-218.

Marshall, I.M. et al., 1994. Predicting the development effort of multimedia courseware. *Information and Software Technology.* **36(5)**: pp. 251-258.

Mikos, R. et al., 1987a. Estimating training effort/cost: A follow-up 'final report' (Part 2). *Performance & Instruction.* **26(7)**: pp. 33-43.

Mikos, R. et al., 1987b. Estimating training effort/cost: A follow-up 'final report' (Part 1). *Performance & Instruction.* **26(5)**: pp. 24-29.

Miles, K., 1990. A generic model for rapid estimation of CBT development time. In: *13th InterAgency/Industry Training Systems Conference, Orlando, FL, November 1990.*

Miles, K.W. and Griffith, E.R., 1993. Developing an hour of CBT: The quick and dirty estimate. *CBT Directions.* **(April-May)**: pp. 28-33.

Milette, M. and Trevor-Deutsch, L., 1995. *Training cost calculation spreadsheet: classroom versus computer-based training.* Visual Solutions Inc.

Orey, M. et al., 1994. Development efficiency and effectiveness of alternative platforms for intelligent tutoring for the mobile subscriber radio-telephone terminal. *Computers and Education.* **22(4)**: pp. 301-313.

Schooley, R.E., 1988. Computer-based training (CBT) cost estimating algorithm for courseware (CEAC). In: *11th Interservice Industry Training Systems Conference,* pp. 319-328.

Senbetta, G., 1991. *An inquiry of time and cost estimating for computer-based training courseware design and development as determined by modified Delphi method.* (Ph.D. thesis, Purdue University).

Senbetta, G., 1992. CBT time and cost estimation: what do the experts say? In: *10th Annual CBT Conference and Exposition,*

Shepperd, M., Schofield, C. and Kitchenham, B., 1996. Effort estimation using analogy. In: *18th International Conference in Software Engineering, Berlin, March 96.*

Steinberg, E., 1984. *Teaching computers to teach.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Tan, W. and Nguyen, A., 1993. *Lifecycle costing models for interactive multimedia systems: Interactive multimedia practice and promise.* London: Kogan Page.

US Navy, 1987. *Request for technical proposal.* RTP (Navy Contract N61339-87-R-2043), US Navy. Quoted in:Miles, K., 1990. A generic model for rapid estimation of CBT development time. In: *13th InterAgency/Industry Training Systems Conference, Orlando, FL, November 1990.*

4.11 ENDNOTES

[1] Jay, Bernstein and Gunderson (1987) used "CBT development cost factors" as a collective description of the factors, variables and characteristics which affected "development time". The paradigm terms have been substituted where appropriate along with use of productivity adjustment factors.

[2] Senbetta (1991) asked his respondents to rank product, client and resource factors according to their impact on their estimates of "CBT time and cost". The top five factors in each group were ranked by the respondents using a Delphic technique. Points were awarded for position in the top five and the result summed to give the overall position in the group. The paradigm terms have been substituted where appropriate along with use of productivity adjustment factors.

[3] Gery (1987) used "factor" and "variable" almost interchangeably to describe productivity adjustment factors which "...individually and collectively contribute to the total time expended in courseware development". She claimed that there were thirty-seven factors but only thirty-six are listed in this and an earlier version (Gery 1986). The paradigm terms have been substituted where appropriate along with use of productivity adjustment factors.

[4] Schooley (1988) used "parameters" to describe productivity adjustment factors. The paradigm terms have been substituted where appropriate along with use of productivity adjustment factors.

[5] Kearsley (1985) used "rules" or "questions" to describe the productivity adjustment factors. The paradigm terms have been substituted where appropriate along with use of productivity adjustment factors.

[6] Miles (1990) used "variables" to describe the productivity adjustment factors. The paradigm terms have been substituted where appropriate along with use of productivity adjustment factors.

[7] Milette and Trevor-Deutsch (1995) used "CBT development weighting factors" to describe one productivity adjustment factor. The paradigm terms have been substituted where appropriate along with use of productivity adjustment factors.

[8] Tan and Nguyen (1993) only use one productivity adjustment factor which they described as "Basic costing assumptions". The paradigm terms have been substituted where appropriate along with use of productivity adjustment factors.

[9] Mikos et al. (1987b) used "Project Complexity Multiplier" to describe productivity adjustment factors. The paradigm terms have been substituted where appropriate along with use of productivity adjustment factors.

[10] Golas (1993) used "variables" to describe the productivity adjustment factors. The paradigm terms have been substituted where appropriate along with use of productivity adjustment factors.

[11] Bergman and Moore (1990) effectively used the "Q" factor as a productivity adjustment factor. The paradigm terms have been substituted where appropriate along with use of productivity adjustment factors.

[12] Marshall et al. (1994) used "cost drivers" and "super cost drivers" to describe productivity adjustment factors. The paradigm terms have been substituted where appropriate along with use of productivity adjustment factors.

[13] Avner (1988) used "production rates" described as "Hours of Effort/Hour". The productivity rate figures are inverted to match the consistent definition presented in this thesis.

[14] Garton et al. (1984) described the effect of the programmer on "speed of development" which they measure using time elapsed from the start to end of the programming period measured in months. They describe, this as "development time", although from the brief description provided, they appear to be measuring

elapsed time as defined in the measurement paradigm. Individual programmers involved in twenty-seven courseware modules were rated in terms of their programming and lesson design skills, subject matter knowledge and ability to work independently to produce a "programmer competency" profile. This potentially interesting study was impossible to convert into the measurement paradigm because only summary results were presented in graphical format. The main findings from this study are presented here for information:

- There is little correlation between length of the program measured in 60 bit bytes, and "development time" (r=0.276)
- There is Little correlation between programmer programming skill rating and "development time" (r=0.648)
- Programmer lesson design skill rating was a significant predictor of "development time"; those lessons programmed by programmers with better lesson design skills took less time to develop (r=0.773)
- Programmer subject matter knowledge rating was a significant predicator of "development time" (r=0.777)
- The more programmers involved in a module, the longer the "development time" required to produce the module (r=0.533)

5. COURSEWARE DEVELOPMENT CASE STUDIES

> "Successful production of multimedia material requires literally hundreds of [developer] hours of planning; hundreds of [developer] hours of programming; hundreds of [developer] hours of observing student behaviour; and hundreds of [developer] hours of revision, revision, revision. Even then, the task is not finished. It must be debugged and made bullet-proof so that students will not become frustrated and give up. Consequently, most people who begin the process of developing multimedia courseware give up soon into the process. It's impossible in the initial heady days, to really comprehend the vast quantity of [effort] that will be required." (Soloman 1994)

As Soloman (1994) indicated the process of developing quality multimedia courseware could consume development effort to the point where non-commercial authors either give up or reduce the interactivity or media content to deliver an inferior product. There have been few attempts to systematically collect and analyse courseware development effort data. Historically, the discipline and dedication required to systematically collect data from projects did not command the same interest as exploring the educational potential of courseware. The net result of the lack of research was to leave the field open to expert opinion about the "mythical development to delivery time ratio" (Marshall et al. 1995b). Those few studies published tended to stop short of a rigorous analysis of the development data, preferring to try to confirm that the *effort to learner time ratio* was similar to the "industry average" (Gailey 1973b; Cates 1994).

In this Chapter, four cases studies will be presented to evaluate the need for a measurement paradigm and to analyse its use in measuring and comparing effort estimation studies. An analysis of productivity adjustment factors as the basis for an effort estimation model will be presented. Two other studies which contained courseware development data were re-analysed to identify any relationships which were missed in the original analysis. These analyse a sole author and small team courseware developments in terms of use of the measurement paradigm and development of effort estimation models. The final case study presents new data collected from a large scale

courseware project developed by a consortium. The case studies will be presented under the following headings:

- An early life-cycle estimation model
- Using the measurement baseline
- Small team historical development
- A large scale courseware development

The first case study presents an analysis of courseware development effort estimation using productivity adjustment factors.

5.1 AN EARLY LIFE-CYCLE ESTIMATION MODEL

One of the problems encountered in multimedia courseware development is the need early on in the courseware development life-cycle for the developer to be able to predict the effort required to undertake the project. There is ample evidence that expert opinion produces neither reliable nor consistent results with the associated risks of not delivering or delivering less than was promised in the original specification (Jay et al. 1987; Baker 1994; Soloman 1994; Canale and Wills 1995; Tennyson et al. 1995). This problem could be reduced if an effort estimation model was used to reliably and constantly predict development effort using data available in the courseware specification phase of the development life-cycle.

The aim of this case study was to investigate the potential of using productivity adjustment factors for effort estimation early on in the development life-cycle. In Chapter 5 seventy seven productivity adjustment factors were identified as possibly affecting multimedia courseware development effort. Data from seventy one courseware development projects were used to investigate the effect of productivity adjustment factors on effort estimation. An early effort estimation model based on productivity adjustment factors was then developed and evaluated.

5.1.1 Objectives

While confidence in and reliability of a general early life-cycle effort estimation model will always be less than for a developer specific model used later in the life-cycle, it does provide a basis for client discussion about the effort associated with different courseware developments. Allowing for these limitations of an early life-cycle estimation model, this case study explores the following.

- The evaluation of the key productivity adjustment factors in differentiating courseware developments
- The synthesis and evaluation of an early life-cycle effort estimation model for courseware development

5.1.2 Methodology

In an ideal world it would be possible to evaluate all seventy seven productivity

adjustment factors because developers would be patient and dedicated enough to collect or describe them for every courseware development. Unfortunately, patience and dedication tend to be in short supply among courseware developers[1]. Even if enough data could be collected, a model which required seventy seven inputs would hardly be practical for courseware development estimators to use.

Key productivity adjustment factors

Heemstra (1992) suggested that only the most dominant productivity adjustment factors need be considered in effort estimation models. This was seen as essential because with seventy seven productivity adjustment factors which could affect courseware development, it would be difficult to take them all in to consideration during effort estimation. Over 1200 different software productivity adjustment factors have been identified by Noth and Kretzschmar (1984) as potentially affecting effort. However, most models use only a subset of between four and twenty to estimate software development effort. It was therefore considered highly likely that most of the seventy seven courseware productivity adjustment factors were highly inter-correlated and could be adequately described by a much smaller set.

Expert opinion was used to reduce the initial set of seventy seven productivity adjustment factors to a more manageable number. In Chapter 5 factors with a final composite ranking (FCR) above the third quartile (Q3=50.58) were identified. Table 5.1 shows the nineteen key factors which met that criteria.

Table 5.1- Nineteen key productivity adjustment factors

Group	Headings	Description of individual productivity adjustment factors	Code
Developer	Team	Experience	DTE
Subject Matter	Content	Complexity	SCC
Course	Interactivity	Conditional branching	CIC
Technical	Authoring tools	Type	TAT
Technical	Media	Graphics, animation and simulation	TMG
Course	Testing	Feedback	CTF
Course	Screen	Design	CSD
Course	Testing	Response analysis	CTR
Course	Instructional method	Courseware complexity	CIC1
Developer	Subject matter experts	Availability	DSA
Developer	Subject matter experts	Experience	DSE
Client	Involvement	Consistency	CIC2
Subject Matter	Behavioural objectives	Level	SBL
Technical	Media	Video and audio	TMV
Developer	Designers	Experience	DDE
Developer	Organisation	Development procedures	DOD
Developer	Manager	Experience	DME
Subject Matter	Content	Existing material	SCE
Client	Involvement	Decision-making	CID

Rating scales

Each of the productivity adjustment factors was given an associated rating scale based on the analysis of similar scales from existing effort estimation methods, research results, discussion with the developers and the results of the survey carried out in Chapter 3. A linear scale with values ranging from zero to four was used. The higher the value the more productive the project. Some of the productivity adjustment factors made use of all five points on the scale while others used only two or three points as appropriate.

In addition to the nineteen key productivity adjustment factors identified the data set was classified in terms of the decade in which the work was carried out, whether the development was commercial or non-commercial and the size of the total courseware project development. Table 5.2 shows the criteria used to translate project conditions into associated productivity adjustment factor ratings.

Table 5.2 - Rating scale for productivity adjustment factors

Productivity adjustment factors	Rating scale				
	5	4	3	2	1
Developer team experience	0 projects	1 project	2 projects	3 projects	4 or more projects
Subject matter content complexity	Complex		Medium		Simple
Course interactivity conditional branching	Adaptive	Highly complex	Complex	Simple	Linear
Technical authoring tools type	Assembly language	General purpose language	Authoring language	Authoring system	Intelligent authoring system
Technical media graphics, animation and simulation	Complex animation/ simulation	Simple animation/ simulation	Complex Graphics	Simple Graphics	Text only
Course testing feedback	Diagnostic with remediation	Diagnostic feedback	Right answer	Right/wrong	None
Course screen design	Windowing	Complex graphics	Simple graphics	Complex text	Simple text
Course testing response analysis	Natural language	Simple free text	Multiple choice	Simple selection	None
Course instructional method courseware complexity	Complex		Medium		Simple
Developer subject matter expert's availability	Restricted contact	Monthly contact	Weekly contact	Daily contact	Unrestricted contact
Developer subject matter expert's experience	0 projects	1 project	2 projects	3 projects	4 or more projects
Client involvement consistency	Inconsistent				Consistent
Subject matter behavioural objectives level	Problem solving	Higher order principles	Lower order principles	Abstract concepts	Concrete concepts
Technical media video and audio	Analogue interactive video	Analogue interactive audio	Digital video	Digital audio	None
Developer designer's experience	0 projects	1 project	2 projects	3 projects	4 or more projects
Developer organisation's development procedures	None		Evolving		Well established
Developer manager's experience	0 projects	1 project	2 projects	3 projects	4 or more projects
Subject matter content existing material	None	Tutor delivered materials	Written materials	Courseware	Multimedia courseware
Client involvement decision-making	Committee (greater or equal to 5)	Committee (Less than 5)	Nominated individual	Client contact	Team member
Age	1960-69	1970-79	1980-89	1990-99	2000-09
Organisational goal	Non-commercial				Commercial
Total project size (Lh)	Less than 1	1 to 10	11 to 20	21 to 40	More than 40

The rating scale works on the basis that the lowest rating applicable is selected when analysing the courseware developed or a project environment.

Data

In this case study data from seventy one courseware developments were available for analysis. This data had been collected over the last three years by the author. It included courseware data from the WinEcon project, published studies, unpublished studies and the MEEM data set (Marshall et al. 1994b). A summary of the projects with references to sources is presented in Appendix 7.2.

5.1.3 Analysis

The data set was entered into Minitab 10 for Windows for detailed statistical analysis. All seventy one projects were analysed as one set before a subset of the forty five projects delivered since 1990 were considered in isolation. While the larger data set was of interest it was hoped that the modern data subset would provide results reflecting current development practices.

Analysis of whole dataset

The initial analysis consisted of investigating each factor's correlation with effort. For those productivity adjustment factors with five rating points, their correlation with effort was used while one-way analysis of variance (ANOVA) was used to investigate those with only two or three rating points. This analysis identified that *learner time (LT) is* highly positively correlated with effort.

Excluding learner time

It was decided to investigate models based on the other productivity adjustment factors before looking at *learner time (LT).* Stepwise regression was used with those productivity adjustment factors which had a significant correlation or ANOVA with effort excluding *learner time.* The following predictive equation explains only a small proportion of the variance (R^2= 0.415).

$$\text{Effort} = -2970.4 + 192\text{SCC} + 674\text{AGE} + 449\text{CSD} \qquad \text{(A)}$$

This was not considered a good model of the process and stepwise regression of the twenty one productivity adjustment factors was carried out excluding *learner time (LT).* The following predictive equation presents the results of stepwise regression excluding

learner time (LT).

$$\text{Effort} = -5579.9 + 884\text{AGE} + 650\text{CSD} + 539 \times \text{SCE} \qquad \text{(B)}$$

While the equation explains slightly more of the variance (R^2= 0.471) it is unlikely to produce a good model for estimating effort.

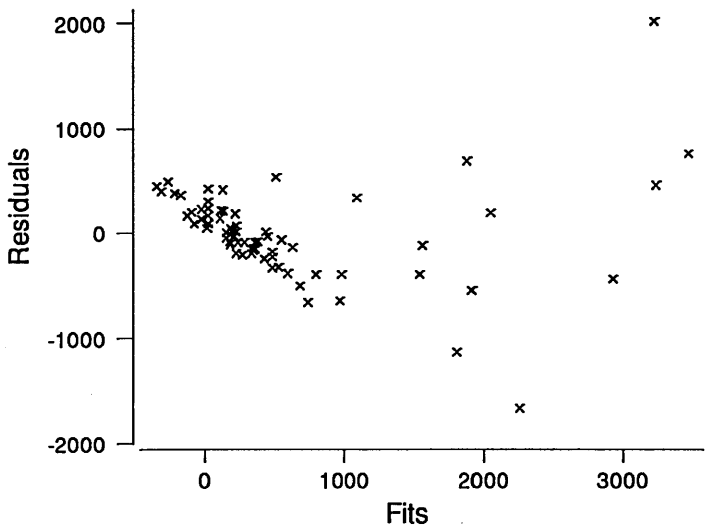
Adding learner time

Having explored the results produced excluding *learner time* stepwise regression was used on all the factors to produce the following predictive equation.

$$\text{Effort} = -3388 + 155\text{LT} + 315\text{SCE} + 592\text{AGE} + 332\text{CSD} - 295\text{DSE} + 249\text{DME} \qquad \text{(C)}$$

Although the resulting equation explains most of the variance (R^2=0.784) and the individual factors are highly significant, the model is not a particularly good one. As Figure 5.1 shows the plot of residuals against fits produced an expanding "cone" effect which is indicating that the variability of effort does not remain constant over the range of predictors considered. The regression model assumes consistency of variance.

Figure 5.1 - Plot of residuals against fits for equation "C"



Omitting the two observations with the largest residuals from the data set increased the variance explained (R^2=0.838). However, this did not improve the quality of the model

which retained the expanding "cone" in its plot of residuals against fits.

Looking at powers of learner time

Earlier analysis of Senbetta's (1991) results indicated that effort may be related to a power of *learner time*. To investigate this possibility the *learner time* was raised to three example powers and correlated with effort. As Table 5.3 shows, the highest correlation of effort is with *learner time* rather than with any power of *learner time*.

Table 5.3 - Correlation of effort with powers of learner time

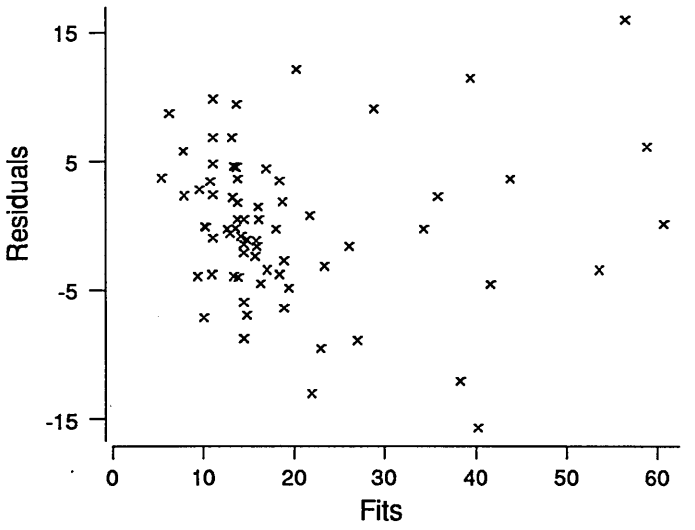
	Effort	Learner time	(Learner time)^2	(Learner time)$^{0.5}$
Learner time	0.727			
(Learner time)^2	0.705	0.975		
(Learner time)$^{0.5}$	0.713	0.983	0.999	
(Learner time)$^{0.67}$	0.691	0.961	0.998	0.995

This suggests that it is unlikely that any better results could be obtained by using a power of *learner time* rather than *learner time* itself. However, using the square root sometimes stabilises the variance. In an attempt to improve the quality of the model the square root of effort was used as the basis for stepwise regression. The resulting equation explained almost as much of the variance as the previous models (R^2=0.823) with all factors significant.

$$\sqrt{\text{Effort}} = -41.1 + 2.10\text{LT} + 9.09\text{AGE} + 5.61\text{CSS} + 4.57\text{SCE} - 4.08\text{DSE} + 3.43\text{DME} \quad \text{(D)}$$

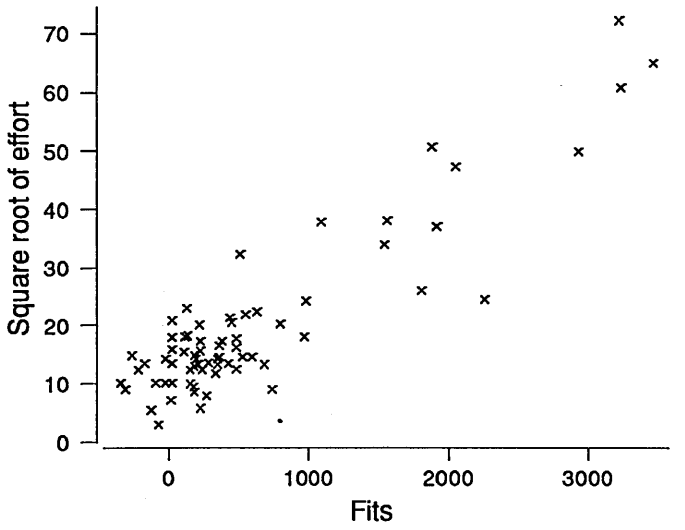
Figure 5.2 shows that the plot of residuals against fits for equation "D" now has a much better "cloud shape" which suggests that the square root of effort has approximately constant variability over the range of predictors considered.

Figure 5.2 - Plot of residuals against fits for equation "D"



This is reflected in Figure 5.3 which shows a plot of the square root of effort against fits for regression equation "D".

Figure 5.3 - Plot of square root of effort against fits for equation "D"



Analysis of modern projects

Since the aim of the investigation was to produce an effort estimation model which could be useful in the future, it was felt that better results could be obtained by concentrating on modern courseware only delivered since 1990. This reduced the data

set to just forty five projects but they appeared to reflect some of the diversity of techniques and technologies identified in Section 3.1 of this Chapter.

The previous analysis was repeated, with similar results, on the data set of modern courseware developments to produce the following predictive equation.

$$\text{Effort} = -902 + 119\text{LT} + 159\text{TMV} + 200\text{CSD} \qquad \text{(E)}$$

All the predictors are significant and can explain most of the variance (R^2=82.6). L*earner time (LT)* on its own accounted for 64.50% of the variance. Technical media video and audio (TMV) improved this by a further 15.06% while course screen design CSD only improved this by a further 3.08%. However, one data point caused some concern because it had both a very large standard residual and a large influence. The problem was tracked down to a project in which the developer had reported effort was very small in comparison to that predicted by the model. Since the courseware had been developed in a non-commercial organisation it was possible that an overly optimistic value for effort had been reported by the developer.

Improving the model

Plotting the residuals against fits for the equation "E" produced an expanding "cone" shape indicating a poor quality model. The square root of effort was calculated and then stepwise regression was used to produce the following predictive equation.

$$\sqrt{\text{Effort}} = 2.95 + 188\text{LT} + 3.64\text{TMV} + 1.58\text{CIC1} \qquad \text{(F)}$$

While the variance explained (R^2=0.767) is slightly less than with equation "E", the constant is no longer significant indicating that the equation could be forced through the origin. In the equation *learner time (LT)* on its own accounted for 55.21% of the variance. Technical media video and audio (TMV) improved this by a further 18.24% while course instructional method complexity (CIC1) only improved this by a further 3.27%. Analysis of plots of the residuals against fits and square root of effort against fits indicated that the model quality was good and that there was no systematic lack of fit.

5.1.4 Evaluation of effort estimation model

While the models based on the whole data set are of general interest, the age of some of the points suggested that it would be more beneficial to evaluate the model using the modern data set.

$$\text{Effort} = (2.95 + 188\text{LT} + 3.64\text{TMV} + 1.58\text{CIC1})^2 \qquad \text{(G)}$$

The evaluation involved making a bootstrap estimate of the error by systematically removing one data point and using regression to predict the missing observation with the remaining observations. Table 5.4 show the results of the bootstrap estimate of predictive accuracy of effort estimation equation "G". The bootstrap estimate of error produced a $\overline{MRE}$ of 1.00 and *PRED(0.25)* of 0.31 or *PRED(0.30)* of 0.38. On this basis the model clearly falls in to the poor category for effort estimation models (Campbell et al. 1988). These $\overline{MRE}$ and *PRED(l)* results almost require the extension of the Campbell, Conte and Rathi's (1988) criteria to include "very poor".

Table 5.4 - Bootstrap estimate of absolute relative error for equation "G"

Project number	Actual effort *(Dh)*	Predictive effort *(Dh)*	Absolute relative error
1	28	74.70	0.17
2	590	839.74	0.42
3	180	397.88	1.21
15	78	307.69	2.94
16	420	258.92	0.38
29	50	88.31	0.77
30	96	217.50	1.27
31	100	148.79	0.49
32	480	281.94	0.41
33	175	165.42	0.06
34	175	196.93	0.12
35	300	166.69	0.44
36	270	182.99	0.32
37	150	146.80	0.02
38	240	168.38	0.30
39	210	218.12	0.04
40	210	267.07	0.27
41	210	273.59	0.30
42	200	179.95	0.10
43	180	207.86	0.15
44	200	118.09	0.41
45	312	208.95	0.33
46	300	220.72	0.26
47	175	110.41	0.37
48	60	160.49	1.67
49	450	239.55	0.47
50	100	148.79	0.49
51	150	187.08	0.25
52	150	119.70	0.20
53	150	132.93	0.11
54	600	933.43	0.56
55	2572	1565.48	0.39
57	531	135.91	0.74
58	323	140.58	0.56
59	331	146.20	0.56
60	237	143.13	0.40
61	1044	445.79	0.57
62	166	292.37	0.76
63	71	199.47	1.81
64	220	184.70	0.16
65	2500	2054.39	0.18
66	8	170.00	20.25
67	155	298.91	0.93
68	260	290.12	0.12
69	405	835.49	1.06

5.1.5 Discussion

Initially this seems a disappointing result. It appears that even with a relatively large data set, a model cannot be produced which meets the minimum criteria for an adequate

courseware effort estimation model. However, there are a number of factors which need to be considered which affect this analysis. Firstly, the data was drawn from a large number of different sources of which only the WinEcon and MEEM data sets attempted to use the standards set out in the measurement paradigm. Using the checklist proposed in Chapter 3 clearly indicates the problems with this data set. As the checklist in Table 5.5 shows, the only criterion that can be guaranteed is that the data is drawn from commercial and non-commercial projects which include sole authors, teams and consortiums. All the other criteria in the checklist, which describe the conditions under which the data must be collected, and have been left blank because there is no guarantee that all the authors were describing their project in exactly the same way. This highlights the weakness in trying to collect together project data to prepare a general effort estimation model without guaranteeing that the data collected and terminology used are consistent.

Secondly, the nineteen productivity adjustment factors and three other factors have a number of weaknesses. The first and most obvious is that the nineteen productivity adjustment factors included may not be the best set to describe modern courseware developments. Some of the sources of expert opinion were slightly dated and may not reflect current practices. The other weakness is the use of a linear rating scale from 1 to 5 for all the factors. It may be that some factors have a larger influence and should be given a greater weighting. In addition, while some of the rating scales meet Boehm and Wolverton's (1980) criteria for effort estimation models, others such as "subject matter content complexity" are clearly not objective and are subject to intra-developer and inter-developer interpretation.

However, the model does have some good points. The results suggest that there is some relationship between effort and *learner time* however it is defined by the individual developers. The final model does provide a method of estimating courseware development effort early on in the development life-cycle by estimating a few critical factors. While confidence in the result is limited, it does provide a starting point for further refinement by the developer and it could form the basis for a more rigorous method.

Table 5.5 - Measurement paradigm checklist for early life-cycle estimation model

Heading	Sub-heading	Description	Yes	No
Organisational context	Commercial	Sole author	√	
		Team	√	
		Consortium	√	
	Non-commercial	Sole author	√	
		Team	√	
		Consortium	√	
Courseware product quality	Status	Finished testing phase and ready for publishing		
		Produced to master copy or camera ready quality		
	Includes	Software which controls the learning environment		
		Computer-based presentation of one or more media		
		Computer-based assessment of learning		
		Learning support materials		
		Installation materials		
		Learner record system		
Educational quality	Mastery score gain	Mean greater than 50%		
	Completion rate	Greater than 95%		
Measurement baseline	Includes	Instructional design phase		
		Multimedia design phase		
		Multimedia development phase		
		Courseware integration phase		
		Testing phase		
	Excludes	Courseware specification phase		
		Maintenance phase		
Development activities (within measurement baseline)	Included effort	Development team members		
		Subject matter experts based with client		
	Excluded effort	Client's employees/learners		
		Non-team employees of developer		
Sizing factors	Learner time	Mean measured from a sample of 30 target learners		
		Estimated by client/developer		
	Screens	Base screens and overlays		
	Interactions	Meaningful and related to course objectives		
	Lessons	Discrete/named subsection of course		
	Objectives	Number of specific objectives		
	Media objects	Number of discrete media objects		
	Source lines of code	Logical source lines of code		
Measures	Elapsed time	Measurement baseline measured in days		
	Development time	Measurement baseline measured in working hours		
	Effort	Measurement baseline effort measured in *Dh*		
	Team size	Total number of developers on the team		
	FTE developers	Full-time equivalent developers on the team		
	Cost	Monetary value of effort and overheads		
	Productivity	Output/input measured in $\overline{Lh\,Dh^{-1}}$		
	ETLTR	Effort to learner time ratio measured in $\overline{Dh\,Lh^{-1}}$		
	DEC	Developer effort charge to client (Currency Dh^{-1})		

5.2 USING THE MEASUREMENT BASELINE

Tragham (1996) in his thesis presented the results of a project to develop multimedia courseware for delivery of an induction course for the University of Abertay Dundee Library. The courseware which used video, audio and extensive graphical design features was developed in Asymetrix Multimedia Toolbook 3.0. The finished courseware, source code and development data were used in this analysis.

5.2.1 Objectives

The library courseware used video, audio, text, graphics and simulation to deliver a learning experience to undergraduates. This courseware development provided an opportunity to explore the following two aspects.

- Evaluation of the extent to which the courseware development data could be collected in the format specified by the measurement paradigm using the measurement baseline
- Evaluation of the educational and product quality elements of the measurement paradigm for multimedia courseware

5.2.2 Reported results

The main aim of the development was to produce multimedia courseware to support student use of the library facilities. Tragham (1996) was sole author/developer for the courseware and two senior members of the library staff acted as the client and contact person for the duration of the project. The courseware was developed in an *elapsed time* of eighty-two days starting on the 17 of August 1995.

As part of the project Tragham (1996) recorded development effort expended under headings representing the six phases of the courseware development life-cycle with two additional headings called "learning" and "other". The "learning" heading covered activities such as learning about the courseware or the subject matter whilst the "other" heading was used to record any effort expended which did not fit into any other classification. Table 5.6 shows the courseware development data recorded by Tragham (1996).

Table 5.6 - Courseware development data (Tregham 1996)

Category	Phase heading	Effort *(Dh)*	Phase elapsed time *(Days)*	Development time *(hours)*	Percentage of effort (%)
Life-cycle	Courseware specification	24.5	66	24.5	7.8
	Instructional design	24.5	42	24.5	7.8
	Multimedia design	111.0	69	111	35.1
	Multimedia development	52.5	48	52.5	16.6
	Integration	2.0	1	2.0	0.6
	Testing	11.0	11	11.0	3.5
Additional	Learning	74.5	55	74.5	23.6
	Other	16.0	8	16.0	5.1
Totals		316	300	316	100.1

Table 5.6 shows *elapsed times* from the start of a particular phase to the end of that phase. The elapsed time for the whole project was only eighty-two days in comparison to the total of three hundred days found by summing the individual phases of *elapsed times*. This discrepancy was caused by the overlapping of development activities during the project. Whilst the courseware development data was recorded using the headings from the measurement paradigm, the development used was rapid prototyping. This can be seen from the courseware specification phase *elapsed time* which stretches from the start of the project to nearly the end. The dominance of learning which accounts for approximately 24% of the recorded development effort can be explained by the fact that Tragham (1996) had no prior experience in programming with Asymetrix Multimedia Toolbook 3.0 although he did have experience of another object-oriented programming language.

5.2.3 Analysis of courseware data

Tragham (1996) collected data during the development but he did not analyse the results in any detail preferring to comment qualitatively on the overall development and problems encountered. However, the finished courseware, supporting documentation and data were available for this study. Table 5.7 shows a summary of the results obtained from the analysis of the library courseware.

Table 5.7 - Summary of library courseware results

Heading	Description	Total course
Sizing factors	Mean *learner time (Lh)*	0.42
	Estimated *learner time (Lh)*	1
	Screens	87
	Interactions	10
	Lessons	1
	Objectives	5
	Media objects	50
	Source lines of code *(SLOC)*	1273
Measures	*Elapsed time (days)*	112
	Development time (h)	346
	Effort *(Dh)*	346
	Team size	2
	FTE developers	1
	Cost (£)	8650
	Productivity ($\overline{Lh\ Dh^{-1}}$)	0.0012
	Effort *to learner time ratio* ($\overline{Dh\ Lh^{-1}}$)	823.81
	Developer effort *charge* ($£\ Dh^{-1}$)	25

Sizing factors

Estimated *learner time* was agreed with the librarians and other people involved in the development. It represented a consensus on the amount of time required by an average undergraduate to complete the material covered in the courseware. As can be seen from Table 5.7, the mean *learner time* for actual student use is much shorter than the estimate. Although the mean *learner time* is measured using a slightly smaller group than proposed in the measurement paradigm this was unlikely to affect the result. All students completed the courseware in under thirty minutes with the shortest time being twenty minutes with the majority of students taking about twenty five minutes (0.42 *Lh*).

The Toolbook source code was analysed to provide counts of screens, interactions, media objects and logical source lines of code. The counts were carried out independently three times over a period of two weeks by the author. Discrepancies between the counts were investigated and re-counted to ensure consistency. The finished courseware met the requirements of the original courseware specification which required that only one lesson should be produced to cover five objectives. An individual screen in the courseware required an average of 17.4 seconds for the learner to view. This compares with an average of 26.1 seconds reported by (Cates 1994) in his novice courseware developer study. The library courseware required 3.98 $Dh\ Screen^{-1}$ to

develop which was very similar to the 4.28 *Dh Screen*$^{-1}$ reported by Cates (1994).

Measures

Table 5.7 shows the total effort and *development time* reported by Tragham (1996) in the development of the courseware and by this author testing and evaluating the results. The testing phase required an additional twenty *Dh* of effort over thirty elapsed days to try out the courseware. This evaluation was carried out using a group of twenty five undergraduates for whom the material was intended. The length of the *elapsed time* reflects the fact that only one computer and courseware CD-ROM were available for testing.

While no-one was charged for the development, a nominal *developer effort charge* of £25 per developer hour was used to estimate the cost. The resulting courseware cost of £8650 represents the nominal charge that would be levied to the library for the development effort expended without taking into account contributions to overheads and other associated costs or a contribution to profit. This particular course is taught to approximately 500 students annually by the library and would replace 50 hours of librarian effort per year. On this basis, the resulting courseware would need to be used for approximately 6 years to recover its development costs providing capital, hardware and recurring costs were ignored.

The reported productivity for this development is considerably lower than Gailey's (1973a). This perhaps reflects the inclusion of media in the courseware and also maybe the complexity of the interface design in the library courseware which is missing from the household electricity course. Gailey also had an advantage in developing more and longer *learner time* modules. Learning about the content or the authoring tool consumed 23.6% of Tragham's effort over one relatively small module whilst Gailey was able to distribute this effort over four modules which were 8.74 times as large as Tragham's. The effect on productivity is therefore quite small. Similarly, the effort *to learner time ratio* produced by the library courseware is large in comparison to Gailey's but in line with the range reported by Marshall, Samson and Dugard (1994a) for multimedia courseware.

5.2.4 Use of the measurement paradigm

As Table 5.9 shows, Tragham's study conforms to the requirements of the measurement paradigm. The major omissions are the lack of learning support materials and a record system. These were intended to be developed as part of the project but were not completed before Tragham's contract finished. The other problem area was that insufficient students undertook the pilot study to meet the criteria described by the measurement paradigm. While this was disappointing, it was unavoidable due to the fact that the students were about to undertake the library induction session as part of their normal course. This would have changed the conditions under which the courseware was used and therefore it was decided not to recruit the last five students to undertake the pilot study. However, all the students who started the courseware pilot finished with a mean mastery gain score of 0.67 which exceeds the criterion laid down in the measurement paradigm.

5.2.5 Discussion

Tragham's (1996) study shows that it is possible to use the measurement baseline as the basis for collecting courseware data. There would appear to be a need for "learning" and "other" headings in addition to the basic six suggested by the measurement baseline. The "learning" heading separates out activities associated with learning about the authoring tool or subject matter. In the case of a novice developer this accounted for a large proportion of the development effort. The "other" heading was used to collect effort data which were difficult to classify under any of the main headings.

The courseware met the educational criteria albeit with less than the required number of students undertaking the pilot study. Using a slightly smaller sample than recommended was unlikely to affect the completion rate, mastery gain score or mean *learner time* produced in the pilot study. Whilst the courseware did meet the educational quality criterion, it failed to meet two elements of the product quality criterion. However, the two missing elements would require relatively little additional effort to ensure that the courseware reached the product quality standard specified in the measurement paradigm. The courseware, although appearing to meet the criteria, suffers from non-commercial quality audio and video. While it was acceptable for the pilot study, the video and audio elements would have to be re-shot to produce professional quality

results. Considering there was only 116 seconds of video, 81.36 seconds of audio and 61 seconds of animation with audio in the finished courseware this would not demand a large amount of effort to recreate using more professional facilities and artists.

Table 5.8 - Measurement paradigm checklist for library courseware

Heading	Sub-heading	Description	Yes	No
Organisational context	Commercial	Sole author		√
		Team		√
		Consortium		√
	Non-commercial	Sole author	√	
		Team		√
		Consortium		√
Courseware product quality	Status	Finished testing phase and ready for publishing	√	
		Produced to master copy or camera ready quality	√	
	Includes	Software which controls the learning environment	√	
		Computer-based presentation of one or more media	√	
		Computer-based assessment of learning	√	
		Learning support materials		√
		Installation materials	√	
		Learner record system		√
Educational quality	Mastery score gain	Mean greater than 50%	√	
	Completion rate	Greater than 95%	√	
Measurement baseline	Includes	Instructional design phase	√	
		Multimedia design phase	√	
		Multimedia development phase	√	
		Courseware integration phase	√	
		Testing phase	√	
	Excludes	Courseware specification phase	√	
		Maintenance phase	√	
Development activities (within measurement baseline)	Included effort	Development team members	√	
		Subject matter experts based with client	√	
	Excluded effort	Client's employees/learners	√	
		Non-team employees of developer	√	
Sizing factors	Learner time	Mean measured from a sample of 30 target learners		√
		Estimated by client/developer	√	
	Screens	Base screens and overlays	√	
	Interactions	Meaningful and related to course objectives	√	
	Lessons	Discrete/named subsection of course	√	
	Objectives	Number of specific objectives	√	
	Media objects	Number of discrete media objects	√	
	Source lines of code	Logical source lines of code	√	
Measures	Elapsed time	Measurement baseline measured in days	√	
	Development time	Measurement baseline measured in working hours	√	
	Effort	Measurement baseline effort measured in *Dh*	√	
	Team size	Total number of developers on the team	√	
	FTE developers	Full-time equivalent developers on the team	√	
	Cost	Monetary value of effort and overheads	√	
	Productivity	Output/input measured in $\overline{Lh\ Dh^{-1}}$	√	
	ETLTR	Effort to learner time ratio measured in $\overline{Dh\ Lh^{-1}}$	√	
	DEC	Developer effort charge to client (Currency Dh^{-1})	√	

5.3 SMALL TEAM HISTORICAL DEVELOPMENT

Gailey (1973a) systematically recorded the *effort* and *learner time*[2] associated with the development of courseware to teach students the basic principles of electricity in the home. The courseware was developed on an IBM 360/50 mainframe computer based at Ohio State University in an authoring language called Coursewriter III. Simple text-based terminals were used to deliver instruction to students who were studying an introductory household equipment course at the University. The courseware was intended to support traditionally delivered instruction and had supporting documentation for student use. In her thesis Gailey (1973a) presented the entire source code of the courseware along with the supporting documentation.

5.3.1 Objectives

Although the courseware data was relatively old, it provided an opportunity to explore three objectives.

- Evaluation of the extent to which the original courseware and development data could be presented in the format specified by the measurement paradigm
- Analysis of correlations between sizing factors and effort for courseware developed without media and complex screen design
- Synthesis and evaluation of potential courseware effort estimation model for a specific small team development environment

The age of the data reduced the reliability of predicting development effort for modern courseware developments based on the results from this case study. However, the argument was that modern courseware with its complexity of design and inclusion of media makes it less suitable for software effort estimation techniques. Without this complexity traditional sizing factors, such as logical source lines of code, would be related to effort.

5.3.2 Reported results

While this case study was primarily of historical interest, the age of the technology made it relatively simple to analyse the authoring language aspects of courseware development without the complexity introduced by media and screen design. Gailey

(1973a) although recording and presenting some of the data, failed to fully explore potential relationships between development effort and the various size factors. Table 5.9 shows the four courseware modules developed by Gailey (1973a).

Table 5.9 - Household electricity courseware modules (Gailey 1973a)

Module	Title	Method	Development order
A	Electrical terms	Tutorial	2
B	Safe use of electricity	Tutorial	1
C	Household circuit game	Simulation/game	4
D	Cost of using electricity	Tutorial	3

Effort to learner time ratios

Gailey's (1973b) primary objective was to establish the *effort* and *learner time* required by recording courseware development and delivery data. The courseware was developed mainly by Gailey (1973b) with the assistance of a team of five courseware consultants and four subject matter experts. The material was evaluated using fifty-three students of whom only twenty-four completed the course. Table 5.10 shows the effort *to learner time ratios* for the four individual modules and the total development calculated using the mean *learner time* for the twenty four students.

Table 5.10 - Household electricity *effort to learner time ratios* (Gailey 1973a)

Module	[Effort (Dh)]	[Mean *learner time* (*Lh*)]	[*Effort to learner time ratio* ($\overline{Dh\ Lh^{-1}}$)]
A	185.51	2.64	134
B	88.00	0.62	142
C	135.59	0.80	168
D	32.30	0.39	83
Total	441.40	4.45	128

Distribution of development tasks

The courseware appears to have been developed with Gailey (1973b) in the combined role of subject matter expert and courseware designer. Coding of the modules was carried out by the other team members who acted as courseware consultants and the subject matter experts provided advice on the content. Gailey (1973b) recorded the effort for the author and other people involved using eleven task headings throughout the courseware development life-cycle. Table 5.11 shows the effort expended during tasks related to writing the content of the individual modules.

Table 5.11 - Effort for content writing related tasks (Gailey 1973b)

Tasks	[Effort (*Dh*)]				
	Module A	**Module B**	**Module C**	**Module D**	**Total**
Research	15.25	4.83	6.67	0.50	27.25
Correspondence	2.67	2.00	2.33	0.33	7.33
Consultation - subject matter	11.75	17.00	14.67	2.84	46.26
Initial writing	38.92	16.92	23.92	7.88	87.64
Modification after use	11.50	3.25	0.50	0.75	16.00
Secretarial assistance	9.67	3.58	8.33	2.00	23.58
Development of manual	10.75	1.25	6.17	1.00	19.70
Sub total	100.51	48.83	62.59	15.30	227.76

Table 5.12 shows the effort consumed in programming related tasks during the development of the modules.

Table 5.12 - Effort for programming related tasks (Gailey 1973b)

Tasks	[Effort (*Dh*)]				
	Module A	**Module B**	**Module C**	**Module D**	**Total**
Consultation - [Courseware]	4.50	3.50	7.00	1.00	16.00
Training in Coursewriter	9.00	0.00	0.50	0.00	9.50
Coding and input	56.50	12.25	52.75	12.50	134.00
Debugging	15.00	23.42	12.75	3.50	54.67
Sub total	85.00	39.17	73.00	17.00	214.17

Distribution of effort

Table 5.13 shows the distribution of effort of the author and other people for the main tasks carried out during the project.

Table 5.13 - Distribution of effort by tasks (Gailey 1973b)

Tasks	Distribution of effort					
	Author		**Other people**		**Total**	
	Dh	**%**	***Dh***	**%**	***Dh***	**%**
Content writing	186.06	68	41.17	19	227.23	46
Computer-related activities	66.67	24	147.50	68	214.17	44
Refinement after operation	21.50	8	28.75	13	50.25	10
Total	274.23	100	217.42	100	491.65	100

As Table 5.13 shows, in this project approximately 10% of the total effort was expended during the "refinement after operation" stage of the courseware development. This highlights the need to ensure that measurements are taken using the same measurement baselines. Exclusion of the effort expended during the testing phase would have apparently improved the productivity but without actually doing so.

Development experience

Gailey (1973b) reported a 32%[3] reduction in the author's *effort to learner time ratio* between the first and third tutorial module produced. This was in agreement with anecdotal evidence that developers are more productive in later modules because they have aready developed support tools and techniques in earlier modules. The change from tutorial to simulation/game mode of delivery resulted in a 60%[4] increase in the *effort to learner time ratio* indicating that there was a need for the author and developers to come to grips with this new technique and develop tools to help support the new type of courseware.

Learner evaluation

Student use of the courseware was evaluated using a questionnaire administered after the course. The attitude of the students towards the courseware was generally positive with 81% of the nineteen respondents recommending continued use of the tutorial modules and 52% favouring the use of the simulation module. Unfortunately, no pre-test or post-test assessment results were published for the students who used this course.

5.3.3 Analysis of courseware data

Whilst Gailey (1973b) collected data during the courseware development she failed to fully analyse relationships between effort and sizing factors other than *learner time*. She also made no attempt to validate a method for estimating development effort from the data available even though with a data set limited to just four data points it would have been a relatively minor task to explore these aspects. Luckily the data required to re-analyse the courseware was available. Table 5.14 shows the results of the analysis of the household electricity courseware.

Table 5.14 - Summary of household electricity courseware results

Heading	Description	Module A	Module B	Module C	Module D	Total course
Sizing factors	Mean *learner time (Lh)*	1.86	0.62	0.80	0.39	3.67
	Estimated *learner time (Lh)*	-	-	-	-	-
	Screens	352	240	224	119	935
	Interactions	163	119	269	50	601
	Lessons	1	1	1	1	4
	Objectives	-	-	-	-	-
	Media objects	NA	NA	NA	NA	NA
	Source lines of code *(SLOC)*	841	516	784	197	2338
Measures	*Elapsed time (days)*	-	-	-	-	-
	Development time (h)	-	-	-	-	-
	Effort *(Dh)*	185.51	88.00	135.59	32.30	441.40
	Team size	9	9	9	9	9
	FTE developers	-	-	-	-	-
	Cost ($)	-	-	-	-	-
	Productivity ($\overline{Lh\ Dh^{-1}}$)	0.0100	0.0070	0.0059	0.0121	0.0073
	Effort *to learner time ratio* ($Dh\ \overline{Lh^{-1}}$)	134	142	168	83	128
	Developer effort *charge* ($\$\ Dh^{-1}$)	-	-	-	-	-

As Table 5.14 shows, the main sizing factors (except for the number of specific objectives and media objects) could be measured using the source code or supporting documentation. The number and nature of the specific objectives were not described in the documentation whilst media objects were not relevant to this courseware. In terms of the measures provided, or which can be derived from this case study, the only major omissions are *elapsed time*, *development time, FTE developers, cost* and *developer* effort *charge*. The omission of costing elements reflected the non-commercial nature of this development and while these could be estimated from some of the other measures, it would have been simpler if they had been reported directly.

Sizing factors

The overview design diagrams, Coursewriter III source code and student manual were presented by Gailey (1973b) in an Appendix to her thesis. Counts of the source lines of code (SLOC), text lines, screens, and interactions were carried out on the source code. Table 5.15 shows a summary of the results of the analysis of the household electricity courseware.

Table 5.15 - Analysis of household electricity courseware (Gailey 1973b)

Module	Development Sequence	Logical SLOC	Text Lines	Number of Screens	Number of Interactions	Effort (Dh)
A	1	841	550	352	163	185.51
B	2	516	814	240	119	88.00
C	4	784	559	224	269	135.59
D	3	197	360	110	50	32.30
Total	-	2338	2283	926	601	441.40

The logical source lines of code (SLOC) represents the number of executable lines of Coursewriter III code in the courseware excluding comments and blank lines. The text lines represents a count of the number of lines of subject matter text presented to the student. Cates (1994) and Yeager's (1987) definition of the total number of screens and interactions delivered by the courseware were used as the basis for both counts.

The general objectives for each module were described in the thesis (Gailey 1973b). Unfortunately, these tended to be stated in the form of very broad aims for each module and although the student manual and source code were provided, neither clearly described the specific objectives to be taught in each module. Analysis of the content and structure of the student manual indicated that each module provided instruction on between three and four specific objectives. This lack of variation would be unlikely to provide any additional information for effort estimation.

Effort estimation

Correlation analysis was used to identify potential relationships between the *learner time*, logical source lines of code, text lines, number of screens, number of interactions and effort. Table 5.16 shows the correlation among the four sizing factors and effort from the household electricity courseware.

Table 5.16 - Correlation among effort and sizing factors

	Effort	*Learner time*	Logical SLOC	Text SLOC	Screens
Learner time	0.905				
Logical SLOC	0.969	0.771			
Text SLOC	0.250	0.057	0.348		
Screens	0.933	0.911	0.867	0.456	
Interactions	0.691	0.336	0.837	0.231	0.461

As Table 5.16 shows, there were large positive correlations between *learner time,* logical SLOC, screens and effort. However, for four observations in the data set only the logical SLOC is significantly correlated with effort ($r \geq 0.9500$ $p \leq 0.05$). Similarly,

there was a large positive correlation between *learner time* and screens but this failed to reach the 5% significance level.

It appears for this courseware that the traditional software effort estimation model could be used to predict courseware development effort. Regression analysis identified a significant relation between effort and logical source lines of code. The following predictive equation describes the relationship between effort and logical source lines of code (R^2=0.938).

$$\text{Effort} = 0.2154 \times \text{logical source lines of code} - 15.529 \qquad \text{(H)}$$

A slightly better explanation of the variance can be achieved by using the following predictive equation (R^2=0.9815).

$$\text{Effort} = 0.0809 \times (\text{logical source lines of code})^{1.1289} \qquad \text{(I)}$$

As Table 5.17 shows, both the equations met the criteria for good effort estimation models (Campbell et al. 1988).

Table 5.17 - Comparison of effort equations

Effort equations	R^2	$\overline{MRE}$	*PRED(0.25)*
A	0.938	0.123	1.00
B	0.982	0.079	1.00

However, the small size of the sample makes it difficult to generalise the relationships to all simple authoring language-based courseware without media. In terms of potential use, substituting estimation of effort for estimation of logical SLOC would probably be viewed as a retrograde step by most courseware development practitioners.

5.3.4 Use of the measurement paradigm

One of the weaknesses of courseware effort estimation methods and, to a certain extent, of software effort estimation models is the difficulty in ensuring that the data used in the development and validation of the models are consistent with that used in their calibration and actual use. The proposed measurement paradigm identifies how data should be collected to ensure consistency and to identify the situations in which a model could be effectively used. Table 5.18 shows the extent to which Gailey's (1973a) data

conforms to the proposed measurement paradigm.

As can be seen from Table 5.18, the household electricity course did conform reasonably well to the measurement paradigm. In terms of the courseware product quality the only area of weakness was the difficulty in telling if the finished courseware had installation instructions. All other aspects of the courseware product quality were clearly identifiable from the documentation, including the use of a learner record system which was used to track student usage. The educational quality aspect of this case study was not well defined. Gailey (1973a) made no reference to the use of pre-tests or post-tests in the evaluation of the courseware. It was therefore impossible to calculate the mastery score gain and it has therefore been left blank. The completion rate for the household electricity was only 45.28% which failed to meet the criterion that 95% of students who started the courseware pilot study should finish. This may indicate that more work was required to make the courseware more interesting and relevant to students but it could also reflect the fact that the courseware was being used in addition to traditional instruction. While students were encouraged to use the courseware, they may not have seen it as an essential element of their course.

From the description of the development it appears that the courseware was produced using a close approximation to the measurement baseline allowing for the fact that there was no multimedia aspect to this courseware. The recorded effort follows the recommendations of the measurement paradigm. However, it is worth noting that because the project was developed internally, there were no client-based subject matter experts which would explain the not applicable (NA) rating for this criterion.

The calculation of *learner time* in the case study is based on only twenty-four learners who completed the course rather than the thirty recommended by the measurement paradigm. However, it was felt that this sample was close enough to that required to accept the mean value of *learner time* reported by Gailey (1973a). There is no indication in the thesis of the developer's original estimate of the *learner time* the courseware was intended to deliver.

5.3.5 Discussion

Gailey's (1973a) household electricity courseware follows the basic principles for data collection and analysis outlined in the measurement paradigm. Use of the measurement paradigm checklist enables the weaknesses in the study to be identified. In this example, the simplicity of the courseware produced the expected result that there was a relationship between development effort and logical source lines of code. The courseware was entirely software-based with no media and very little screen design which may explain the results. Surprisingly, there was no significant relationship between the number of lines of text, screens and interactions and effort. But with only four points a large positive correlation would be required to achieve significance which is a function of sample size and as well as estimated correlation.

Table 5.18 - Measurement paradigm checklist for household electricity courseware

Heading	Sub-heading	Description	Yes	No
Organisational context	Commercial	Sole author		√
		Team		√
		Consortium		√
	Non-commercial	Sole author		√
		Team	√	
		Consortium		√
Courseware product quality	Status	Finished testing phase and ready for publishing	√	
		Produced to master copy or camera ready quality	√	
	Includes	Software which controls the learning environment	√	
		Computer-based presentation of one or more media	√	
		Computer-based assessment of learning	√	
		Learning support materials	√	
		Installation materials		√
		Learner record system	√	
Educational quality	Mastery score gain	Mean greater than 50%	?	?
	Completion rate	Greater than 95%		√
Measurement baseline	Includes	Instructional design phase	√	
		Multimedia design phase	√	
		Multimedia development phase	√	
		Courseware integration phase	√	
		Testing phase	√	
	Excludes	Courseware specification phase	√	
		Maintenance phase	√	
Development activities (within measurement baseline)	Included effort	Development team members	√	
		Subject matter experts based with client	NA	NA
	Excluded effort	Client's employees/learners	√	
		Non-team employees of developer	√	
Sizing factors	Learner time	Mean measured from a sample of 30 target learners	√	
		Estimated by client/developer		√
	Screens	Base screens and overlays	√	
	Interactions	Meaningful and related to course objectives	√	
	Lessons	Discrete/named subsection of course	√	
	Objectives	Number of specific objectives		√
	Media objects	Number of discrete media objects	NA	NA
	Source lines of code	Logical source lines of code	√	
Measures	Elapsed time	Measurement baseline measured in days		√
	Development time	Measurement baseline measured in working hours		√
	Effort	Measurement baseline effort measured in *Dh*	√	
	Team size	Total number of developers on the team	√	
	FTE developers	Full-time equivalent developers on the team		√
	Cost	Monetary value of effort and overheads		√
	Productivity	Output/input measured in $\overline{Lh\ Dh^{-1}}$	√	
	ETLTR	Effort to learner time ratio measured in $\overline{Dh\ Lh^{-1}}$	√	
	DEC	Developer effort charge to client (Currency Dh^{-1})		√

5.4 A LARGE SCALE COURSEWARE DEVELOPMENT

The WinEcon courseware project was funded under the United Kingdom Higher Education Funding Council's (HEFC) Teaching and Learning Technologies Programme (TLTP) (HEFC 1992; HEFC 1993). Eight universities collaborated on the project which was co-ordinated by the Centre for Computing in Economics at Bristol University. The consortium's aim was to produce Microsoft Windows-based courseware to replace tutorials and augment lectures for approximately 12000 students per academic year studying 'Introductory Economics' courses at UK universities (Sloman 1995). Each of WinEcon's 25 chapters of courseware made extensive use of text, graphics, simulation and student interaction to deliver on average between three and five hours of student activity. In addition, the finished courseware was customisable to meet the requirements of individual lecturers. Although the WinEcon courseware was supplied at nominal cost to HEFC-supported institutions in the UK, it has been sold to commercial organisations and overseas higher education establishments and has also won a number of prizes for its educational quality.

5.4.1 Objectives

This case study used the finished courseware, project management documents and developer interviews to explore the following two aspects.

- Evaluation of the extent to which the courseware development data could be collected in the format specified by the measurement paradigm for a large courseware project
- Evaluation of the use of sizing factors and code-based measures for multimedia courseware effort estimation models

5.4.2 Overview of project

The project was originally intended to start in October 1992 with the completed courseware to be delivered in September 1994 for the first non-pilot study student use in October 1994. During this twenty-four month period the consortium members were expected to design and develop commercial quality courseware from a outline specification which they had submitted to the HEFC to win the funding bid. The core development team, based at Bristol, were responsible for project co-ordination, overall

design and the provision of development tools. To ensure commitment to the project, each of the consortium members were allocated individual courseware chapters to develop along with the financial resources to employ courseware development teams.

Courseware development teams

The consortium members tended to use experienced undergraduate economics educators as authors and team managers. These individuals had varying levels of experience of courseware production and were mainly selected due to their involvement with the original funding bid. The courseware developers employed by the consortium contained a mixture of experienced and inexperienced developers most of whom had not previously worked together. These courseware developers were mainly young graduates employed on fixed term contracts for the duration of the development.

Courseware authoring language

WinEcon courseware was developed in Asymetrix ToolBook, the object-oriented authoring language based around a book metaphor (Price and Hobbs 1994; Price and Hobbs 1995). Pages form the basis of the high level structure while objects (including pages) can have OpenScript programs attached. OpenScript is a fully functional object-oriented programming language. The finished courseware used the ToolBook Runtime to deliver the courseware. This runs the courseware but prevents students from editing pages or changing 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 book metaphor was extended to describe the modules which made up the courseware with a related collection of pages being described as a chapter.

Project management

The overall project management was co-ordinated by the Centre for Computing in Economics at Bristol who reported to the WinEcon Executive Committee drawn from consortium and associate members. Each consortium member was allocated a collection of chapters to produce and the financial resources to enable the development to take place. The sub-contracting of chapters to the consortium members was organised on a contractual basis with the issue of well defined briefs with agreed deliverables and

review dates. Quality standards and code reviews were carried out by the team at Bristol.

The twenty-six chapters planned in the original WinEcon submission to HEFC were reduced to twenty-five following a re-analysis of the content. Chapter nine was dropped but the original chapter numberings were retained. Entering the original project plan and resources into Microsoft Project 4 for Windows indicated that only one of the teams had sufficient resources to carry out its tasks; all the other teams were over-allocated. Using the resource levelling to remove the over-allocation estimated the project finish date to be in late October 1996. While resource levelling was a sub-optimal technique it did indicate that the original project plan was over-ambitious.

Delivery problems

The most obvious problems encountered in the actual delivery of the project was that it overran the planned end of the contract by twelve months and that the courseware released in October 1995 had reduced functionality in comparison to the original specification. For example, the student record keeping system was missing although this was included in a maintenance upgrade launched in December 1995. The original bid documentation had also suggested the inclusion of extensive video resources. This was dropped quite quickly due to the cost and the perceived difficulty in guaranteeing adequate delivery platforms for student use in client universities. This suggest that the estimated delivery date of late October 1996 produced by resource levelling was perhaps not to far out if the video elements had been retained.

Courseware delivered

The first public release of the WinEcon courseware consisted of 163346 lines of source code comprising 25 chapters organised as 693 courseware pages. Each page delivered up to 0.1 hours of *learner time* and each chapter on average provided 2.77 hours of student activity. WinEcon made extensive use of text, graphics, simulation and student interaction to deliver the material. To meet the requirements of individual lecturers, WinEcon was 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.

5.4.3 Collection of courseware data

This analysis of the courseware development data was based upon effort collection questionnaires, project records and documentation. Data was also collected by means of interviews and discussions with the software development team and the WinEcon project manager over a period of ten days. Five of the days of the investigation were expended prior to the public launch of WinEcon and five of the days just before the second release of the courseware. The software development manager was an active participant in the data collection exercise and its subsequent analysis.

Development measurements

Project records, documentation and questionnaires were used to collect development data for each of the chapters produced by the consortium. Table 5.19 shows the development data for each WinEcon chapter along with the team responsible for the development. Chapter 9 is missing from the list because it was amalgamated into other chapters during implementation.

Table 5.19 - Summary of development data collected from WinEcon project teams

Chapter number	Team number	Effort *(Dh)*	*Development time (hours)*	*FTE Developers*	*Team size*
1	7	350	280	1.25	4
2	8	672	448	1.50	4
3	3	700	560	1.25	4
4	5	3360	2240	1.50	5
5	1	1225	980	1.25	1
6	1	1225	980	1.25	1
7	3	2100	1680	1.25	3
8	5	1890	1260	1.50	5
10	7	1050	840	1.25	7
11	8	1680	1120	1.50	8
12	2	1470	980	1.50	2
13	2	1470	980	1.50	2
14	2	1470	980	1.50	2
15	7	1400	1120	1.25	7
16	8	1260	840	1.50	8
17	7	1400	1120	1.25	7
18	8	2184	1680	1.30	8
19	3	2100	1680	1.25	3
20	1	1225	816.67	1.50	1
21	8	420	280	1.50	8
22	5	3150	2100	1.50	5
23	1	700	560	1.25	1
24	3	1050	840	1.25	3
25	6	1050	840	1.25	6
26	6	1050	840	1.25	6

Sizing factor measures

The finished WinEcon source code, executables and project documentation were analysed to count the number of objectives, screens, interactions and media objects. These sizing factors were counted three times over a period of a month to reduce the variability of the results and any discrepancies were re-analysed. Table 5.20 shows the sizing factors for the WinEcon courseware.

Table 5.20 - Sizing factors for WinEcon courseware

Chapter number	Learner hours	Logical SLOC	Media Objects	Screens	Interactions	Objectives
1	0.6	1780	7	56	5	1
2	3.4	11215	46	254	17	6
3	2.1	3665	38	113	7	8
4	4.7	10159	51	298	13	7
5	2.5	4089	32	145	10	6
6	3.1	3659	19	177	10	4
7	2.6	3839	23	136	13	9
8	2.9	9375	39	385	26	7
10	2.1	4461	19	121	6	4
11	2.6	6224	20	187	24	6
12	3.5	4995	23	234	34	7
13	4.3	3626	27	332	50	7
14	4.4	5134	30	332	6	6
15	2.8	6313	26	210	27	5
16	3.3	10326	30	264	45	6
17	1.5	4850	24	96	6	3
18	3.4	6263	59	207	30	6
19	3.6	5097	24	209	12	8
20	1.3	1587	8	52	3	6
21	2.3	3683	15	170	12	6
22	4	8094	12	336	27	6
23	2.1	3202	6	145	20	4
24	2.9	4741	13	199	46	9
25	1.5	3270	14	67	2	5
26	1.8	5195	15	81	5	8

Code-based measures

Potential code-based measurements were identified in collaboration with the software development manager who had extensive ToolBook experience. An automated measurement tool was then constructed to collect 53 measures from the first release of the WinEcon program source code (Price et al. 1996). The measurements were grouped under three main headings:

- WinEcon specific
- ToolBook specific
- Portable

The first group of five WinEcon specific measurements were unique to this development. They represented coding constructs developed as part of the core development team's overall design philosophy. As such, they were unlikely to be useful outside the consortium. Table 5.21 describes the WinEcon specific measurements.

Table 5.21 - Description of the five WinEcon specific measurements

Measurement	Description
WinEcon-text-cards	WinEcon step by step text cards
WinEcon-prof-fields	WinEcon typographically standard help fields
WinEcon-popups	WinEcon popup groups
WinEcon-graphs	In-house graph 'objects'. These are actually a collection of numerous objects, some grouped, some not
WinEcon-spreadsheets	WinEcon spreadsheet windows floating over the page and linked via DLL

The second group of ten ToolBook specific measurements described OpenScript language or page metaphor specific features. While they were described in ToolBook terms, they could be extended to other authoring systems or languages. Table 5.22 describes the ToolBook specific measurements.

Table 5.22 - Description of the eleven 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

The final group of thirty-seven portable measurements described features which were not ToolBook specific. Table 5.23 describes the portable measurements in OpenScript terms; these may have equivalents in other authoring or programming languages.

Table 5.23 - Description of the thirty-seven portable measurements

Measurement	Description
page-count	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.
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 scripts
dos	'do' statements in scripts
sends	'send' statements in scripts. This is similar to 'procedures' in C
to-gets	Function definition
to-handles	Handler definition 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

5.4.4 Analysis of data

Predicting effort, and hence cost, is one of the major concerns of most measurement programmes. However, so far attempts to predict effort for courseware projects have not been very successful (Marshall et al. 1995a) so anything that could be learned from the WinEcon project would be very valuable. Table 5.24 provides a summary of the WinEcon sizing factors and measures.

Table 5.24 - Summary of WinEcon courseware results

Heading	Description	Total course
Sizing factors	Mean *learner time (Lh)*	
	Estimated *learner time (Lh)*	69.7
	Screens	4113
	Interactions	456
	Lessons	25
	Objectives	150
	Media objects	620
	Source lines of code *(SLOC)*	134,844
Measures	*Elapsed time (days)*	1170
	Development time (h)	26,045
	Effort *(Dh)*	35,651
	Team size	111
	FTE developers	34.05
	Cost (£)	1,000,000
	Productivity ($\overline{Lh\ Dh^{-1}}$)	0.0020
	Effort *to learner time ratio* ($Dh\ \overline{Lh^{-1}}$)	511
	Developer effort *charge* ($£\ Dh^{-1}$)	16.83

The data collected from the WinEcon courseware development was analysed using the sizing factors and the code-based measurements.

Analysis of sizing factors

Estimates of the magnitude of sizing factors could be made early on in the courseware development life-cycle. It was therefore important to identify any relationships that existed between effort and sizing factors. Table 5.25 shows the correlation between effort and the sizing factors obtained from the WinEcon courseware.

Table 5.25 - Correlations between effort and the WinEcon sizing factors

	Effort	Media Objects	Screens	Interactions	Objectives	Learner time	SLOC
Media objects	0.383						
Screens	0.516	0.449					
Interactions	0.193	0.166	0.592				
Objectives	0.337	0.276	0.295	0.324			
Learner time	0.635	0.534	0.839	0.509	0.449		
SLOC	0.466	0.618	0.675	0.365	0.227	0.580	
Pages	0.635	0.534	0.839	0.509	0.449	1.000	0.580

As Table 5.27 shows, only screens, SLOC, *learner time* and pages were significantly positively correlated with effort ($r \geq 0.3961$, $p \leq 0.05$). However, pages screens, SLOC and *learner time* were also significantly positively correlated with each other. Stepwise regression on all the sizing factors identified the following predicative equation.

$$Effort = 23.8 + 6.49\, pages\ (C)$$

While it was highly significant it explained only about 40% of the variance (R^2=0.404) and the addition of the other sizing factors in various combinations did not 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 *pages* 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. Table 5.26 shows the bootstrap estimate of predictive accuracy for the effort equation.

Table 5.26 - Bootstrap estimate of predicted effort accuracy

WinEcon Chapter	Actual effort *(Dh)*	Estimated effort *(Dh)*	Absolute relative error
1	50	66.50	0.33
2	96	253.20	1.64
3	100	163.80	0.64
4	480	294.22	0.39
5	175	186.57	0.07
6	175	227.32	0.30
7	300	187.95	0.37
8	270	209.57	0.22
10	150	160.73	0.07
11	240	190.52	0.21
12	210	253.62	0.21
13	210	316.98	0.51
14	210	326.12	0.55
15	200	205.77	0.03
16	180	241.10	0.34
17	200	112.06	0.44
18	312	240.50	0.23
19	300	254.40	0.15
20	175	98.63	0.44
21	60	178.89	1.98
22	450	265.03	0.41
23	100	163.80	0.64
24	150	214.65	0.43
25	150	117.84	0.21
26	150	139.86	0.07

The bootstrap estimate of absolute relative error produced a $\overline{MRE}$ of 0.435 and a *PRED(0.25)* of 0.40 which placed it in the poor model category (Campbell et al. 1988). While the result was disappointing, the predictive equation does have the advantage that it could be used early on in the development life-cycle. Even in the earliest WinEcon courseware specification and design documentation, the course was described in terms of chapters, pages and screens. This effort estimation equation could certainly be used for future developments which adopt the WinEcon design philosophy.

Analysis of code-based measures

Analysis involved producing predictive equations using multiple regression of effort with subsets of the available measures. The subsets were defined in three ways: *a priori* considerations, best predictors, and the best portable predictors.

A priori considerations

Research from software measures indicated that the size and complexity of the product influence the effort consumed in its production (Boehm et al. 1994). Four of the measures relate to size; *raw-lines* of code, number of *objects*, number of *named-objects*, and number of *pages*. Table 5.27 shows the correlations between effort and the size measures grouping.

Table 5.27 - Correlations between effort and the size measures grouping

	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

There were four traditional measures related to complexity (Oviedo 1993); *ifs, elses, conditions,* and *whens*. Other measures related to complexity, which were also considered appropriate in this context, were number of *propertied-objects*, *WinEcon-graphs* and *steps*. Table 5.28 shows the correlations between effort and the complexity measures.

Table 5.28 - Correlations between effort and the complexity measures grouping

	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

Programming style was also felt to influence effort with more expert or talented programmers being able to economise on effort by using approaches which made good use of the authoring tool. The following measures and combinations of measures were related to different aspects of programming style:

- system variables; heavy use of these suggested weak programming
- *to-get+to-set;* use of these suggested the programmer was more experienced
- *pagescript-raw-lines/raw-lines;* centralized versus decentralized code distribution
- *propertied-objects/named-objects;* manipulated objects versus static screen elements
- *scripted-objects/pagescript-raw-lines;* code density of objects

Table 5.29 describes the correlation coefficients between effort and the style measures grouping.

Table 5.29 - Correlations between effort and the style measures grouping

	effort	system-variables	to-get+to-set	pagescript-raw-lines/ raw-lines	propertied-objects/named-objects
system-variables	-0.039				
to-get+to-set	0.110	0.375			
pagescript-raw/raw-lines	-0.063	-0.121	-0.117		
propertied-objects/ named-objects	-0.189	0.139	-0.278	-0.169	
scripted-objects/ pagescript-raw-lines	-0.183	-0.095	-0.136	-0.818	0.265

The use of stepwise regression on sixteen potential predictors with only twenty-five observations was considered unreliable. In order to reduce the number of potential predictors, measures with large correlations with effort but small correlations with each other were selected from each grouping. Unfortunately, all the size measures were highly correlated with effort but also with each other. *Pages* had the highest correlation with effort but *named-objects*, the next highest, was also highly correlated with *pages*. *Pages* and *objects* are selected from the size measure group.

Of the complexity measures, the first five were highly correlated with effort and the first four were highly correlated among themselves. *Propertied-objects* were also highly correlated with three of the first four; this was selected along with *elses* which has an insignificant correlation with *propertied-objects*. None of the style measures had a high correlation with effort but the best of them, *propertied-objects/named-objects,* was selected.

A stepwise regression on these five measures suggested that only *pages* and *elses* made a significant contribution to prediction, and the regression on these two accounted for only 55% of the variance of effort. Though the regression is highly significant, it was

unlikely to give any more useful predictions than the previous equation based on only *pages*. Other choices of subsets from these sixteen measures gave similar or worse results. Choosing a subset of measures for effort prediction on *a priori* grounds did not give a usable result. It was reasonable to assume that size, complexity and programmer style would have a significant effect on the development effort. Unfortunately, the analysis suggested that the dependence of effort on these measures was not close enough to produce a useful prediction equation. However, substantial re-coding was required due to the upgrading of ToolBook from version 1.5 to 3 and this may have affected the results.

The best subset of measures

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

$$Effort = 299 + (17.6 \times user\_properties) + (60.2 \times scrolling\_fields)$$

Adding *forwards* to *user-properties* gave an improvement that was nearly significant at 5%, but this did not significantly improve the result of the regression on *user-properties* and *scrolling-fields*. To estimate the accuracy of prediction on new cases, a bootstrap estimate of error was made. The results are shown in Table 5.30.

Table 5.30 - Bootstrap estimate of predicted effort accuracy

WinEcon Chapter	Effort *(Dh)*	Predicted effort *(Dh)*	Absolute relative error
1	350	914.49	1.62
2	672	953.67	0.42
3	700	1003.13	0.43
4	3360	3021.90	0.10
5	1225	910.25	0.26
6	1225	1234.32	0.08
7	2100	1793.01	0.15
8	1890	1531.26	0.19
10	1050	1440.04	0.37
11	1680	1302.62	0.22
12	1470	2242.49	0.52
13	1470	1726.51	0.17
14	1470	1863.11	0.27
15	1400	1299.17	0.07
16	1260	1549.41	0.23
17	1400	674.60	0.52
18	2184	1491.08	0.32
19	2100	2462.63	0.17
20	1225	650.28	0.47
21	420	814.91	0.94
22	3150	2633.65	0.16
23	700	705.43	0.08
24	1050	1225.52	0.17
25	1050	982.62	0.06
26	1050	982.62	0.06

This potentially usable predictor of effort produced a $\overline{MRE}$ of 0.32 and a *PRED(0.30)* of 0.64 which, although still classified as a poor model, was close to the adequate model criterion (Campbell et al. 1988). However, the resulting prediction equation was considered difficult to generalise to other authoring tools or languages. Although *scrolling-fields* was a portable measure, *user-properties* was ToolBook specific and, to a certain degree WinEcon project specific, which reduced the portability of the effort predictor equation.

While this equation produced useful effort predictions for ToolBook-based courseware, the problem still remained of how to estimate both of these measures at an early stage in the development in terms that an average project manager would understand. Analysis of the role of *user-properties* in the code indicated that they were used to track properties resulting from step cards, graphics and other 'house style' objects. Similarly, *scrolling-fields* were not heavily used throughout the chapters because of style guidelines but when they were used, they tended to link together textual information

with examples or simulations. These seemed to indirectly indicate that a substantial programming effort was required to create the example or simulation.

The best subset of portable measures

The method used above was repeated but the analysis was limited to only the group of portable measures. The best of these was *graphics* which had a large positive correlation with effort (r = 0.707). Regression on this alone accounted for 50% of the variance of effort. Once again, only the *elses* measure gave a significant improvement at the 5% level while *words* was nearly significant. Both of these were added to the regression on *graphics*, and *words* only just failed to reach significance in the presence of both the others. Regression on all three accounts produced the following effort prediction equation which accounted for about 69% of the variance of effort.

$$Effort = 161 + (10.4 \times graphics) + (5.37 \times elses) - (0.0516 \times words)$$

Once again, a bootstrap estimate of error was obtained by omitting each observation in turn. The results are shown in Table 5.31.

Table 5.31 - Bootstrap estimates of predicted effort accuracy

WinEcon chapter	Actual effort *(Dh)*	Predicted effort *(Dh)*	Absolute relative error
1	350	618.75	0.77
2	672	1134.64	0.69
3	700	963.05	0.38
4	3360	2785.02	0.17
5	1225	1160.56	0.05
6	1225	1209.14	0.01
7	2100	1484.83	0.29
8	1890	2241.73	0.19
10	1050	1390.13	0.32
11	1680	987.91	0.41
12	1470	2004.90	0.36
13	1470	1379.26	0.06
14	1470	1790.62	0.22
15	1400	1272.93	0.09
16	1260	758.67	0.40
17	1400	1220.86	0.13
18	2184	1471.21	0.33
19	2100	2246.05	0.07
20	1225	606.64	0.50
21	420	971.02	1.31
22	3150	2516.70	0.20
23	700	702.94	0.00
24	1050	2106.54	1.01
25	1050	972.42	0.07
26	1050	1651.51	0.57

The bootstrap estimates produced a $\overline{MRE}$ of 0.344 and *PRED(0.30)* of 0.52 which is clearly in the poor category for effort prediction models (Campbell et al. 1988). Again, the results are potentially usable and were certainly better than the results obtained by expert estimators predicting development effort. However, they were not as good as those produced by the ToolBook specific equation.

An effort prediction equation which was able to explain 69% of variance, while not good in software metrics terms, was undoubtedly better than existing courseware effort estimation methods (Marshall et al. 1995a). The equation made 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 was perhaps not surprising given that it was 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 effort consuming activities. Similarly, analysis of the use of *else* in the code indicated 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. Similarly, the *word* measure gave a fair indication of the volume of textual information presented on the pages. Since in the WinEcon courseware all text on the pages was entered by the programmers, this again was a fair indicator of effort.

5.4.5 Use of the measurement paradigm

As Table 5.32 shows, the WinEcon study conforms reasonably well to the requirements of the measurement paradigm. The major omissions were the lack of learning support materials and a student record system. These were intended to be developed as part of the original project but were not completed before the launch date; however, they were released three months later. It is difficult to judge if a sufficient number of students undertook the pilot studies to meet the criteria described by the measurement paradigm. With approximately seventy hours of *learner time* to be evaluated, small sub-sections of the course were piloted individually. The design philosophy encouraged the use of a pick and mix approach to its use with the result that lecturers selected individual chapters and pages that matched the course they wished to teach. It was therefore difficult to guarantee that every chapter and page met the criteria laid down by the measurement paradigm. However, the results from the WinEcon pilot studies were encouraging. For example, MacDonald (1996) piloted WinEcon with first year Economic undergraduates. All the students who started the pilot study finished which made the completion rate 100%. Unfortunately, in this case, a mean mastery gain score of only 0.22 was reported which did not meet the criterion laid down in the measurement paradigm. However, the students selected for this particular pilot study all had A-level Economics which affected the pre-test results. A control group studying in parallel without using WinEcon achieved a very similar mastery gain score. This highlights the need to ensure that an appropriate mixture of students takes part in the pilot study.

5.4.6 Discussion

The WinEcon project overran its original delivery date by some eighteen months or by 75% in terms of the planned elapsed time. It was 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 represented a real courseware development which lasted more than a few months. The project not only yielded courseware development data but 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.

Table 5.32 - Measurement paradigm checklist for WinEcon courseware

Heading	Sub-heading	Description	Yes	No
Organisational context	Commercial	Sole author		√
		Team		√
		Consortium	√	
	Non-commercial	Sole author		√
		Team		√
		Consortium		√
Courseware product quality	Status	Finished testing phase and ready for publishing	√	
		Produced to master copy or camera ready quality	√	
	Includes	Software which controls the learning environment	√	
		Computer-based presentation of one or more media	√	
		Computer-based assessment of learning	√	
		Learning support materials		√
		Installation materials	√	
		Learner record system		√
Educational quality	Mastery score gain	Mean greater than 50%		√
	Completion rate	Greater than 95%	√	
Measurement baseline	Includes	Instructional design phase	√	
		Multimedia design phase	√	
		Multimedia development phase	√	
		Courseware integration phase	√	
		Testing phase	√	
	Excludes	Courseware specification phase	√	
		Maintenance phase	√	
Development activities (within measurement baseline)	Included effort	Development team members	√	
		Subject matter experts based with client	√	
	Excluded effort	Client's employees/learners	√	
		Non-team employees of developer	√	
Sizing factors	Learner time	Mean measured from a sample of 30 target learners		√
		Estimated by client/developer	√	
	Screens	Base screens and overlays	√	
	Interactions	Meaningful and related to course objectives	√	
	Lessons	Discrete/named subsection of course	√	
	Objectives	Number of specific objectives	√	
	Media objects	Number of discrete media objects	√	
	Source lines of code	Logical source lines of code	√	
Measures	Elapsed time	Measurement baseline measured in days	√	
	Development time	Measurement baseline measured in working hours	√	
	Effort	Measurement baseline effort measured in *Dh*	√	
	Team size	Total number of developers on the team	√	
	FTE developers	Full-time equivalent developers on the team	√	
	Cost	Monetary value of effort and overheads	√	
	Productivity	Output/input measured in $\overline{Lh\ Dh^{-1}}$	√	
	ETLTR	Effort to learner time ratio measured in $\overline{Dh\ Lh^{-1}}$	√	
	DEC	Developer effort charge to client (Currency Dh^{-1})	√	

5.5 SUMMARY

In this Chapter four individual case studies have been used to explore various aspects of the measurement paradigm, productivity adjustment factors and courseware development effort estimation models.

The first case study analysed the potential to produce a early life-cycle effort estimation model using data and productivity adjustment factors from seventy one courseware developments. Unfortunately, it was impossible to guarantee that all the data provided used the measurement paradigm. This is reflected in the disappointing results for effort estimation models which were classified as poor (Campbell et al. 1988). This case study also highlighted the problems associated with trying to produce a general effort estimation model based on data collection from a wide range of different sources. While overall the resulting model was poor and the analysis suggested that effort is related to *learner time,* it did provide a simple method of estimating development effort.

The second case study investigated the feasibility of collecting data and measuring courseware development effort using the measurement paradigm. The measurement base-line used a waterfall analogy to describe the relationship between the various phases. Analysis of the data collected indicated that the courseware had been developed using a rapid prototyping approach. The use of the phases described by the measurement baseline enabled the data to be accurately classified and recorded whatever the underlying life-cycle model. In addition, the other aspects of the measurement paradigm simplified the collection and discussion of development process. Unfortunately, with only one development there was insufficient data to produce an effort estimation model for this case study. However, this case study also indicated that for data collection purposes other heading may be required, such as "learning" and "other" to ensure all the effort is collected.

The third case study investigated a historical courseware development as the basis for evaluating courseware effort estimation in a homogeneous small team environment. Gailey (1973a) had systematically collected data as part of the research for her thesis but had only carried out a basic analysis of the results in terms of *effort to learner time ratios*. The advantage of this case study was that its lack of media objects enabled the

code-based measures to be evaluated. Without the media, good results were obtained using *logical source lines of code* as the basis of an effort estimation model. Unfortunately, the use of *logical source lines of code* as the basis of an effort estimation model would probably be unacceptable to the modern developer community and because the development of media is likely to dominate the expenditure of effort in multimedia courseware.

The final case study provided an opportunity to analyse recent large scale development by a consortium of non-commercial developers. Code and other measures were collected, analysed and evaluated using the measurement paradigm. The sizing factors and code-based measures all produced effort estimation models which at best almost reached the adequate criterion (Campbell et al. 1988).

Each of the case studies provided support for the use of the measurement paradigm and did simplify the collection and analysis of courseware effort estimation data and a consistent way to describe what is being measured and how it is measured. Without this rigour there is very little hope of developing good effort courseware estimation models. While the models produced by the individual case studies are typically poor to adequate, they do provide a basis for further development.

5.6 REFERENCES

Baker, J., 1994. One man and his dog. *Interact.* **1(3)**: pp. 16-17.

Boehm, B. et al., 1994. *Cost models for future software life cycle processes: COCOMO 2.0.* (Annals of Software Engineering Draft 2.1), 21 September 1994, Los Angeles, CA: University of Southern California.

Boehm, B.W. and Wolverton, R.W., 1980. Software cost modelling: Some lessons learned. *Journal of Systems and Software.* **1(3)**: pp. 195-201.

Campbell, R.L., Conte, S.D. and Rathi, M.K., 1988. *Early prediction of software size and effort.* Technical Report (SERC-TR-10-P), Purdue University.

Canale, R. and Wills, S., 1995. Producing professional multimedia: project management issues. *British Journal of Educational Technology.* **26(2)**: pp. 84-93.

Cates, W.M., 1994. Estimating the time required to produce computer-based instructional lessons: Descriptive analyses of the production data of novice instructional developers. *Journal of Educational Computing Research.* **10(1)**: pp. 29-40.

Gailey, F., 1973a. An analysis of development: Use time ratios for a computer assisted instruction unit on basic household electricity. In: *Association for the Development of Computer-based Instructional Systems, Ann Arbor, Michigan, 7-9 August 1973.* pp. 1-8.

Gailey, F.H., 1973b. *An analysis of development/use time for a computer-assisted instruction unit on basic household electricity.* (PhD thesis, Ohio State University).

Heemstra, F.J., 1992. Software cost estimation. *Information and Software Technology.* **14(10)**: pp. 627-639.

HEFC, 1992. *TLTP - Phase I.* 1992, Higher Education Funding Council.

HEFC, 1993. *TLTP - Phase II.* Higher Education Funding Council.

Jay, J., Bernstein, K. and Gunderson, S., 1987. *Estimating computer-based training development times.* ARI Technical Report (765), October 1987, Alexandria, VA:

Research Institute for Behavioural and Social Sciences.

MacDonald, Z., 1996. *Personal communication.* (E-mail to I.M. Marshall, 20 April 1996) Leicester, UK: Leicester University.

Marshall, I.M., Samson, W.B. and Dugard, P.I., 1994a. Estimating multimedia development effort for open access education. In: *AETT 94 Computer Assisted and Open Access Education, Edinburgh, UK,*

Marshall, I.M., Samson, W.B. and Dugard, P.I., 1995a. Multimedia courseware. Never mind the quality, how much will it cost to develop? *Association for Learning Technology Journal.* **3(1)**: pp. 110-117.

Marshall, I.M. et al., 1995b. The mythical development to learner time ratio. *Computers and Education.* **25(3)**: pp. 113-122.

Marshall, I.M. et al., 1994b. Predicting the development effort of multimedia courseware. *Information and Software Technology.* **36(5)**: pp. 251-258.

Noth, T. and Kretzschmar, M., 1984. *Estimation of software development projects.* Berlin: Springer-Verlag.

Oviedo, E.I., 1993. Control flow, data flow and program complexity. In: M. Sheppers, Editor *Software engineering metrics volume 1: measures and validations.* London: McGraw-Hill Book Company, pp. 52-65.

Price, S., Cheah, L. and Hobbs, P., 1996. Software quality in end-user programming and object based rapid application development (RAD). In: *BCS Software Quality Management Conference, Cambridge,*

Price, S. and Hobbs, P., 1994. ToolBook 1.5. *The Economic Journal.* **4(425)**

Price, S. and Hobbs, P., 1995. *Asymetric ToolBook 3.0 - Implications for Developers.* Technical June 1994, University of Bristol, Bristol: Centre for Computing in Economics.

Senbetta, G., 1991. *An inquiry of time and cost estimating for computer-based training courseware design and development as determined by modified Delphi method.* (Ph.D. thesis, Purdue University).

Sloman, J., 1995. WinEcon software review. *Economic Journal.* **5(429)**

Soloman, M.B., 1994. What's wrong with multimedia in higher education? *Technological Horizons in Education Journal.* **21(7)**: pp. 81-83.

Tennyson, R.D. et al., 1995. Employment of system dynamics in modelling of instructional design (ISD). In: R.D. Tennyson and A.E. Barron, Editors *Automating instructional design: computer-based development and delivery tools.* Berlin: Springer-Verlag, pp. 603-609.

TLTP Economics Consortium, 1993. *Module descriptors: Details of WinEcon software and modules.* Project report (PH40-13), May 1993, University of Bristol, Bristol: Centre for Computing in Economics.

Tregham, N.E.Y., 1996. *Courseware metrics a business perspective.* (MSc thesis, University of Abertay Dundee).

Yeager, R., 1987. *Personal communication.* (Letter to J. Jay, January 1987) USA: Intercom Inc. Quoted in: Jay, J., Bernstein, K. and Gunderson, S., 1987. *Estimating computer-based training development times.* ARI Technical Report (765), October 1987, Alexandria, VA: Research Institute for Behavioural and Social Sciences.

5.7 ENDNOTES

[1] In an earlier study a questionnaire which included all potential productivity adjustment factors was used. Only two courseware developers from two hundred contacted completed the ten page questionnaire with the personal assistance of the author.

[2] Gailey (1973a) used manpower hours recorded as "development time" measured in hours, student "terminal use time" measured in hours and "development time to student use time" to describe her results. The measurement paradigm descriptions have been substituted.

[3] I was unable to reproduce this value from the data. My values are 47%-49% for author only and 35%-38%-for total development to learner time.

[4] I was unable to reproduce this value from the data. My values are 40%-56% for author only and 25%-41% for total development to learner time.

6. EVALUATION

> "Developers of instructional computer lessons face a difficult question daily: How much [effort] do I need to allocate for the creation of a lesson? Anyone who has faced this question, particularly those of us who have to base budget planning on the answer, recognise how important the 'right' answer is." (Cates 1994)

Over the last thirty years getting the "right" answer as far as courseware developers were concerned had more to do with the effectiveness of the instructional method or quality of the learning experience than with development *effort* and *cost*. However, in the last five years increasing budgetary constraints in education and training have shifted the emphasis on to the consideration of the cost and benefits of multimedia courseware as a method of reducing delivery cost whilst at the same time maintaining adequate quality. (MacFarlane 1992).

There is a growing awareness that multimedia courseware is a global business (Puttnam 1996). As an industry, it lacks the tools and techniques to estimate and manage the production of multimedia courseware. It appears to have no way to estimate the effort required to create the product or to evaluate the effectiveness of the development process. This thesis attempted to establish a measurement paradigm to support the development of effort estimation models which would assist in the development of an engineering approach to courseware development.

This final chapter attempts to draw this thesis to a conclusion under the following headings.

- Critical review
- Future directions for research
- Conclusion

6.1 CRITICAL REVIEW

In this Section the results of the research presented in this thesis will be critically reviewed. The opportunity will be taken to highlight the areas where the author feels an original contribution to knowledge has been made.

6.1.1 Courseware effort estimation methods

Chapter 2 was effectively an extensive literature review of courseware development effort estimation research. In reviewing the research the following areas were considered.

Definition of multimedia courseware

- ***Multimedia definition.*** In a rapidly moving field such as multimedia, any definition must be tied to the advancement of technology. At present, the definition covers most of the technologies which would be described as multimedia but it should be periodically reviewed to ensure its currency.
- ***Courseware definition.*** This area is more complex and is fraught with problems. This is partially to do with the definition of courseware as something which has an "instructional purpose". The definition is subject to interpretation and is wide enough to encompass software-based products from entertainment, information and educational sectors. However, the main aim of this thesis was to look at courseware which would deliver education or training objectives.

Criteria for evaluating the accuracy of estimates

- ***Campbell, Conte, and Rathi's (1988) criteria.*** The use of $\overline{MRE}$ and *PRED(l)*, while widely used in software engineering, have not been used to evaluate courseware effort estimation models. The rationale for using them was based on the results of small scale estimation experiments, surveys and discussion with experts on the accuracy of their estimates. While it makes a lot of intuitive sense to use these criteria, at the present early stage of courseware development effort modelling there may need to be another threshold to divide the "poor" category into "promising" and "poor".

Review of effort estimation methods

- ***Scope of review.*** While the review of the literature was extensive, it was worrying that the total extent of published research and knowledge on courseware effort estimation amounted to no more than twenty papers excluding those published by the author. However, while some quoted or cited papers and publications were unobtainable, there is no reason to believe there are any major omissions.
- ***Relevance of methods.*** The relevance of some of the methods presented can be questioned on the grounds of both age and application area. It is likely that any effort estimation method published before 1990 will not be directly applicable today without major revision. However, the aim was to show what had been done and to show alternative methods for deriving courseware development effort. The same rationale applies to some of the methods which have a wider application area than just courseware.
- ***Conversion of terminology.*** It is always problematic to convert methods to a new terminology because errors and mistakes or meanings not intended by the original author may be introduced. However, it was felt in this case that the use of inconsistent terminology made analysis more difficult and concealed problems.

Evaluation of existing methods

- ***Use of Boehm and Wolverton's (1980) criteria.*** The age and relevance of this particular study can be criticised. While there are more modern criteria for evaluating software effort estimation models, they cover the same ground as this brief seminal paper. On the relevance issue, no comparable work could be found for courseware effort estimation models.
- ***Extent of quantitative evaluation.*** It would have been interesting to extensively evaluate the effort estimates produced by existing methods. However, the lack of information about the context in which the models were developed reduced the scope for a more quantitative evaluation.

Original contribution to knowledge

The original contribution to knowledge in Chapter 2 concerns the identification and structuring of existing methods using a new classification scheme. In addition, the conversion of these methods into the proposed measurement paradigm terminology has enabled cross-method comparison and the synthesis of a more objective method for evaluating courseware effort estimation models

6.1.2 Establishing a measurement paradigm

Chapter 3 defined the basic terminology and established the measurement paradigm for courseware effort estimation. In reviewing the research the following areas were considered.

Expert opinion on courseware development effort

- ***Sample size.*** The small size of the sample and how well it represents the population as a whole causes concern. Only thirty six individuals responded from six hundred and twenty paper and electronic questionnaires despatched. This represents only a 5.81% response rate which is lower than hoped and, in terms of the whole population, those who responded are obviously self-selected and may not accurately represent the total population.
- ***Analysis.*** The analysis of the respondent data was based around the subjective identification of productivity adjustment factor ratings. Unfortunately, most of the results produced were not significant suggesting that these productivity adjustment factors were not adequately discriminating.

The measurement paradigm

- ***The need for a paradigm.*** The confusion in use of terms and terminology which was highlighted in the previous Chapter necessitated the development of the measurement paradigm. Without it, it is impossible to collect or analyse data. It therefore forms the basis for on-going research.
- ***Completeness of the paradigm.*** While considerable care was taken in the attempt to fully define all relevant measurement terminology and to describe the conditions under which effort should be measured, it is only with extensive field research that limitations will be identified.

Underlying assumptions

- ***Courseware product quality.*** One of the major underlying assumptions of the measurement paradigm is that the courseware is being developed to commercial quality. The reason for this is to enable well-defined deliverables to be described which ensures consistent courseware product quality and to clearly define when measurement of effort should stop. Unfortunately, discussions with commercial developers indicated that sometimes they don't deliver all of the products in certain contracts.
- ***Use of the waterfall model.*** The use of the waterfall model is perhaps the weakest assumption underlying the measurement paradigm. Increasingly developers are adopting a rapid prototyping approach for courseware development where the relevance of the waterfall model comes into question. However, practical experience suggests that the most beneficial aspects of the waterfall model were the use of phase descriptions for data collection and the definition of project start and end points based on these phases. It can be argued that these aspects can be translated to any development model and that the waterfall model is just a convenient method of describing these concepts.
- ***Sizing measures.*** Perhaps the weakest area in the measurement paradigm is the lack of a well-defined and objective size measure to describe the basic quantity of courseware to be produced. Each of the alternative size measures explored in this Chapter has its own problems of definition, estimation and measurement. This inevitably results in all existing courseware effort estimation models making use of *learner time*. While it has a well documented set of difficulties, it does have the advantage that it is widely used in the industry and by potential clients.

Original contribution to knowledge

The original contribution to knowledge in Chapter 3 concerns the synthesis of the measurement paradigm. In addition, existing research into courseware development measurement related to effort estimation has been uniquely structured and evaluated.

6.1.3 Productivity adjustment factors

Chapter 4 defined the productivity adjustment factors which may affect courseware development effort and which form the basis for the subsequent creation of effort estimation models. In reviewing the research the following areas were considered.

Classification of productivity adjustment factors

- ***Subjectivity of the process.*** While great care was taken and an iterative approach used to structure and classify the individual productivity adjustment factors, it was still a subjective process. Had more time been available it would have been preferable to employ more developers and to try to reach a wider consensus on both the groupings and headings. Similarly, the classification of the individual productivity adjustment factors into groupings and headings may not be unique.
- ***Completeness of structure.*** The productivity adjustment factors form a classification tree structure in which siblings are related but describe different aspects of the parent heading. It is clear from a brief analysis that not every potential sibling leaf appears on the tree. By using only existing surveys into expert opinion of effort estimation it is unlikely that all the potential productivity adjustment factors have been identified.
- ***Underlying surveys and studies.*** Some of the studies used in identifying the productivity adjustment factors were quite old and their relevance to modern courseware development practices is questionable. However, within the time and financial constraints available the collection of more recent data of productivity adjustment factors was impossible.

Identification of key productivity adjustment factors

- ***Expert ranking.*** The results from the various studies effectively made use of expert ranking to identify the key productivity adjustment factors. While this has the advantage of simplicity, it can result in the experts describing productivity adjustment factors which cause problems in current projects rather than in general terms. Without confirmation from development data, the general relevance of any ranking may be questioned. It would have been more effective to have used some form of Delphi approach with a group of experts rather than to depend on surveys and studies.

- ***Key factor cut off point.*** The use of the third quartile as the cut off point for identification as a key productivity adjustment factor appears arbitrary. The reason for its selection was primarily on the grounds of ease of use, repeatability and the fact that it produced less than twenty key productivity adjustment factors.

Original contribution to knowledge

The original contribution to knowledge in Chapter 4 concerns the synthesis of the productivity adjustment factor classification scheme and identification of key productivity adjustment factors.

6.1.4 Courseware development case studies

Chapter 5 investigated five aspects of the courseware development effort estimation using case studies. In reviewing the research the following areas were considered.

Early courseware effort estimation method

- ***Relevance of data.*** This study demonstrated the danger of a train-spotter's approach to data collection. Collecting data just because it is available is not a good way to systematically gather information. It was also doubtful that every developer was using a measurement paradigm to collect or describe the data.
- ***Rating scales.*** The rating scales were primarily linear and may not accurately reflect the range or diversity of values required to adequately describe courseware productivity adjustment factors.

Small team historical development

- ***Relevance.*** At first sight using data from courseware developed over twenty three years ago would have little relevance to modern courseware effort estimation modelling. However, it did provide the opportunity to explore courseware without the complexity of media objects. The results are relevant to developers who use an authoring language such as Asymetrix Toolbook to create traditional tutorials.

Sole author multimedia development

- ***Relevance.*** The aim of the study was to explore the use of the measurement paradigm to collect and analyse courseware development effort. In that sense the study achieved its aim but with only one courseware development to analyse it was not possible to produce a predictive equation.

Large scale courseware development

- ***Educational quality.*** Unfortunately, it was not possible to evaluate the educational quality of the total course. This was primarily due to the size of the total course and the design philosophy which encouraged individual lecturers to select pages to match the requirement of the course to be delivered.

Original contribution to knowledge

The original contribution to knowledge in Chapter 5 concerns validation of various aspects of the measurement paradigm and the initial development and evaluation of courseware effort estimation models.

6.2 FUTURE DIRECTIONS FOR RESEARCH

Following on from the start made in this thesis there are a number of areas for further research.

6.2.1 Evaluation of the measurement paradigm

The thesis describes a measurement paradigm and evaluates a few aspects through the use of cases studies. However, the value of the measurement paradigm will only become clear if it is adopted by developers and researchers as the standard way to communicate data, ideas and model results. There are early signs that this adoption has began and that the evaluation and further refinement of the paradigm will involve the wider multimedia courseware developer community.

6.2.2 Investigation of size measures

The thesis identifies a number of potential size measures but the case studies only partially investigated their relationship with effort. There is a need to more fully investigate potential size measures using courseware development data.

6.2.3 Productivity adjustment factors

Another area of study would be to carry out an investigation to extend the current list of productivity adjustment factors and to identify key factors based on the knowledge of experts. In addition, the rating scales could be improved through the use of Delphi techniques to achieve a consensus.

6.2.4 Extension of models

At the moment the effort estimation models are intended to produce just effort. There are, however, ample opportunities to explore other aspects such as *development time*. It is also possible to explore the use of risk analysis in the context of courseware effort estimation. Given the range of values associated with *learner time,* it is perhaps better to produce a range of development effort estimations using risk analysis techniques.

6.3 CONCLUSION

The original aim of this thesis was to systematically investigate and evaluate courseware effort estimation methods, measures and models. The approach taken to achieve this aim realised the following objectives.

- Evaluation of existing courseware effort estimation methods against well-defined model criteria
- Synthesis and evaluation of an original measurement paradigm to standardise courseware effort estimation data collection and evaluation
- Synthesis and evaluation of productivity adjustment factors which affect courseware effort estimation using an original classification scheme
- Synthesis and evaluation of courseware effort estimation models using courseware development data

Chapter 2 indicated that the first objective was achieved. Evaluation of the existing courseware effort estimation methods showed that they failed to meet Boehm and Wolverton's (1980) criteria for software effort estimation models. Earlier research indicated that the existing terminology used to describe courseware development effort was inconsistent. Inconsistencies and problems with the effort estimation methods were more easily identified by the use of the consistent terminology proposed by the measurement paradigm.

The results presented in Chapters 3 and 5 indicated that the second objectives had been partially achieved. An original measurement paradigm has been established which attempts to standardise courseware development effort measures and data collection. The measurement paradigm has been described and partially evaluated by the use of the case studies presented in Chapter 6. However, the constraints and limitations identified in Section 6.1.2 of this Chapter apply.

The third objective has been achieved as shown by the results presented in Chapters 4 and 5. While there is some concern about the relevance of using just expert opinion based on surveys and studies collected over the last ten years, it represents the first attempt to systematically identify productivity adjustment factors which affect multimedia courseware development effort. The results of the analysis indicated that courseware effort estimation models based on productivity adjustment factors were not able to produce an adequate model. Evaluation of the effort estimation models which focused on productivity adjustment factors suggested that there were problems with the data used in the evaluation and potentially the rating scales used. However disappointing the results, they nevertheless form the basis for further research.

The final objective has been achieved through the cases studies presented in Chapter 5. Again, while the results of some of the effort estimation models produced were disappointing, a number of important points were highlighted. The first and perhaps most important was that the use of a rigorous estimation method does appear to improve the reported accuracy of effort estimation models. It was also of interest that, despite the problems associated with *learner time,* there was a significant relationship between it and effort as reported by the developers.

The small team study based on historical development indicated that a model centred on logical source lines of code was capable of producing good courseware effort estimation models. While the age of this study may limit its relevance to modern courseware developments because later studies do not demonstrate a relationship with logical source lines of code, it may nevertheless, be relevant to developers who use an authoring language to develop courseware without media. The use of code-based and other models developed in the WinEcon case study, while not producing good models, point towards a direction for further research.

This thesis presents the results of a systematic attempt to evaluate courseware effort estimation methods and establish a measurement paradigm to assist with the collection of data and the development of effort estimation models. However, while it proposes a structure and carries out some initial evaluation it does not represent the last word on the topic but merely the first in a new area for measurement practice and theory within and development of an engineering approach to courseware development.

6.4 REFERENCES

Boehm, B.W. and Wolverton, R.W., 1980. Software cost modelling: Some lessons learned. *Journal of Systems and Software.* **1(3)**: pp. 195-201.

Campbell, R.L., Conte, S.D. and Rathi, M.K., 1988. *Early prediction of software size and effort.* Technical Report (SERC-TR-10-P), Purdue University.

Cates, W.M., 1994. Estimating the time required to produce computer-based instructional lessons: Descriptive analyses of the production data of novice instructional developers. *Journal of Educational Computing Research.* **10(1)**: pp. 29-40.

MacFarlane, A.G.J., 1992. *Teaching and learning in an expanded higher education system: Report including the appendices.* Working Party Report (ISBN 0 9519377 15), December 1992, Committee of Scottish University Principals.

Puttnam, D., 1996. *Presentation to Parliamentary University Group.* 27 February 1996, 16 Great College Street, London SW1P 3RX: Parliamentary University Group.

7. APPENDICES

7.1 RULES OF THUMB

7.1.1 Total development time

- Nothing will happen in the first 3 to 6 months after you decide to go with [courseware]. It doesn't make any difference whether you go with a vendor or start producing your own in house. (Lee and Zemke 1987)
- It takes 200 hours to produce 1 hour of [Courseware] - including all work in all phases. (Mikos et al. 1987)

7.1.2 Team members

- Where team members are not used to working together or are geographically apart, add 10-15% [to the total effort]. (Casey et al. 1988)
- The first course produced by a new [courseware] development group will be a collection of mistakes. Throw it away. (Lee and Zemke 1987)

7.1.3 Analysis and design

- Analysis and design comprise 50% of total effort (Casey et al. 1988)
- Data gathering usually takes longer than expected, especially when there is no indication as to potential sources is given by client. (Casey et al. 1988)
- Poorly defined content will slow down the training effort by about 30-50%. (Casey et al. 1988)
- 50% of total design time is used for analysis/definition. (Casey et al. 1988)

7.1.4 Authoring

- Even a skilled Instructional Design author will revise plus or minus half the material after first or second draft (and then 20-25% in the third draft). (Casey et al. 1988)

7.1.5 Methods

- Realistic case studies are difficult and time consuming to develop. (Casey et al. 1988)

7.2 COURSEWARE DEVELOPMENT QUESTIONNAIRE

University of Abertay Dundee
Department of Mathematical & Computer Sciences
Bell Street, Dundee DD1 1HG, United Kingdom
Tel. +44 (0) 1382-308608
Fax +44 (0) 1382-308877
E-Mail I.MARSHALL@TAY.AC.UK

Multimedia Courseware Development Effort Estimation Questionnaire

This questionnaire is designed to collect data on the factors which influence the effort of developing multimedia courseware. The aim of the research is to produce a general multimedia courseware estimation method. All information will be treated in confidence. Detailed individual or organisational results will not be published or disclosed to third parties. If you are not sure of any technical terms please leave the question blank.

Organisational Background

How would you describe your organisation? (Tick only one) ☐ Commercial ☐ Non-commercial

How many years has your organisation been developing multimedia courseware? ☐☐☐☐ years
How many years has your organisation been developing other types of courseware? ☐☐☐☐ years

What percentage of your organisation's work is related to multimedia or computer-based courseware development?
☐ Less than 1% ☐ 1% to 33% ☐ 34% to 66% ☐ 67% to 100%

Which type(s) of computer-based courseware does your organisation have experience of developing (Tick all that apply)

Single medium (Text and Simple Graphics)
- ☐ Single Medium - Drill and Practice
- ☐ Single Medium - Certification Tests
- ☐ Single Medium - Adaptive Tests
- ☐ Single Medium - Tutorials
- ☐ Single Medium - Intelligent Tutoring Systems
- ☐ Single Medium - Exploratory Environments
- ☐ Single Medium - Low Fidelity Simulations

Multimedia (Video, Audio, Complex Graphics or Animation etc)
- ☐ Multimedia - Drill and Practice
- ☐ Multimedia - Certification Test
- ☐ Multimedia - Adaptive Tests
- ☐ Multimedia - Tutorials
- ☐ Multimedia - Intelligent Tutoring Systems
- ☐ Multimedia - Exploratory Environments
- ☐ Multimedia - High Fidelity Simulations

Describe any other type of computer-based or multimedia products your organisation has developed.

Which of the following media types have been used by your organisation in courseware production? (Tick all that apply).
- ☐ Text
- ☐ Low resolution graphics
- ☐ High resolution graphics
- ☐ Photo-realistic images
- ☐ High fidelity animation
- ☐ Digital video
- ☐ Complex realistic simulation
- ☐ Analogue (interactive) audio
- ☐ Low fidelity animation
- ☐ Digital audio
- ☐ Simple representational simulation
- ☐ Analogue (interactive) video
- ☐ Virtual reality
- ☐ Other (Specify below)

What percentage (%) of the total courseware developed by your organisation is created using?

Assembly language or machine code	☐☐☐ %
General purpose programming languages (eg C, C++, BASIC etc)	☐☐☐ %
Authoring languages which require some programming skills (eg PILOT, TENCORE, HyperCard etc)	☐☐☐ %
Intelligent authoring systems (eg ID2 Expert etc)	☐☐☐ %
Other	☐☐☐ %

List the main authoring tools used below

Effort Estimation Methods

Does your organisation use a rigorous method for estimating multimedia courseware development effort? ☐ Yes ☐ No

If your answered 'Yes' to the previous question, briefly describe how your organisation estimates development effort.

How accurate is your organisation at estimating multimedia courseware development effort?

On average, estimated 'effort' are within	☐☐☐☐ %	of actual estimate
The worst ever under budget project was	☐☐☐☐ %	under estimate
The worst ever over budget project was	☐☐☐☐ %	above estimate

What are the major problems in accurately multimedia courseware development costs?

When estimating the development effort of a multimedia courseware development, do you base your initial estimate on?

☐ Number of expected student hours	☐ Number of screens	☐ Number of frames
☐ Number of lessons	☐ Number of interactions in total	☐ Number of media objects
☐ Number of objectives	☐ Number of interactions per hour	☐ Other (Specify below)

What other key factor(s) influence your initial estimate?

Which of the following do you include in the estimates of the effort of a multimedia courseware development?

☐ Preparation of estimate	☐ Client meetings	☐ Revisions
☐ Training needs analysis	☐ Video production	☐ Management
☐ Multimedia design	☐ Audio production	☐ Copyright clearance
☐ Courseware maintenance	☐ Animation production	☐ Other support
☐ Preparation of support materials	☐ Graphics production	☐ Technical reports
☐ Programming costs	☐ Media integration	☐ Other (Specify below)
☐ Learning about subject matter	☐ Secretarial support	

Estimate the range of development time your organisation would require to develop the following types of courseware.

	Single media (1 hour of learner time)		Multimedia (1 hour of learner time)	
	Minimum	Maximum	Minimum	Maximum
Drill and Practice	☐☐☐☐ hours	☐☐☐☐ hours	☐☐☐☐ hours	☐☐☐☐ hours
Certificate Test	☐☐☐☐ hours	☐☐☐☐ hours	☐☐☐☐ hours	☐☐☐☐ hours
Tutorial	☐☐☐☐ hours	☐☐☐☐ hours	☐☐☐☐ hours	☐☐☐☐ hours
Adaptive Test	☐☐☐☐ hours	☐☐☐☐ hours	☐☐☐☐ hours	☐☐☐☐ hours
Intelligent Tutoring System	☐☐☐☐ hours	☐☐☐☐ hours	☐☐☐☐ hours	☐☐☐☐ hours
Exploratory Environment	☐☐☐☐ hours	☐☐☐☐ hours	☐☐☐☐ hours	☐☐☐☐ hours
Low Fidelity Simulation	☐☐☐☐ hours	☐☐☐☐ hours	☐☐☐☐ hours	☐☐☐☐ hours
High Fidelity Simulation	☐☐☐☐ hours	☐☐☐☐ hours	☐☐☐☐ hours	☐☐☐☐ hours

Based on your experience, estimate the percentage (%) of time spent during a computer-based or multimedia courseware development on each of the following phases.

Initial analysis	☐☐☐☐ %
Overall courseware design	☐☐☐☐ %
Media design	☐☐☐☐ %
Software programming	☐☐☐☐ %
Media creation/sourcing	☐☐☐☐ %
Software design	☐☐☐☐ %
Media/software integration	☐☐☐☐ %
Courseware evaluation	☐☐☐☐ %
Testing/revision	☐☐☐☐ %

How does your organisation describe the 'cost' of a multimedia courseware development?

☐ Monetary value ☐ Duration in developer days ☐ Number of developers involved ☐ Other (Specify below

How many computer-based or multimedia courseware development projects have you been involved in over the last 5 years? ☐☐☐☐ projects

How many computer-based or multimedia projects have you prepared estimates for in the last 5 years? ☐☐☐☐ projects

Which one of the following best describes your role in multimedia development projects over the last 5 years? (Tick one only)

- ☐ Sole author and developer on 'not for profit' projects
- ☐ Sole author and developer on 'for profit' projects
- ☐ Member of 'not for profit' project teams
- ☐ Member of 'for profit' project teams
- ☐ Manager of 'not for profit' project teams
- ☐ Manager of 'for profit' project teams

What is the typical size of the multimedia project development team you have been involved with over the last 5 years? ☐☐☐☐ people

Any other comments on your organisation's cost estimation methods.

Other Comments

Please use the space provided below to note down any other information which you wish to make available about estimating multimedia courseware development effort or cost.

Your Details (Optional)

Name	
Position	
Organisation	
Address	
Town/City	
Country	
Postcode	
Telephone	
Fax	
E-mail	

Please contact me, I would be willing to

- ☐ Collect multimedia courseware data
- ☐ Provide historical multimedia courseware development data
- ☐ Provide opinions on estimating methods/models
- ☐ Validate methods/models against multimedia courseware development data

Please return it to:

I. M. Marshall,
University of Abertay Dundee,
School of Informatics,
Bell Street, Dundee DD1 1HG, United Kingdom.
Fax 44(0)382-308877
E-mail I.MARSHALL@TAY.AC.UK

Thank you for completing this questionnaire.

7.3 DATA SETS

Table 7.1 - Productivity adjustment factors data set

Project	Project Description	DTE	SCC	CIC	TAT	TMG	CTF	CSS	CTR	CIC1	DSA	DSE	CIC2	SBL	TMV	DDE	DOD	DME	SCE	CID	Total	Dave	Subj	Cour	Tech	Cle	Org	AGE	SIZE	Learner time	Effort
1	Linear CAT	5	1	1	2	3	2	4	4	1	2	5	1	3	1	5	5	5	3	1	54	27	7	12	6	2	5	2	5	0.167	28
2	Branching IV	4	3	2	2	3	3	4	4	3	4	4	1	3	4	4	3	4	5	1	61	23	11	16	9	2	5	2	4	3	590
3	MOPS	5	1	3	2	2	4	3	4	5	4	5	1	3	1	5	5	5	3	3	64	29	7	19	5	4	1	2	4	2	180
4	Simple CBT	5	1	2	3	1	3	1	3	3	3	5	1	1	1	5	3	5	5	4	55	26	7	12	5	5	1	3	4	1	400
5	Simple CBT	4	1	2	3	1	3	1	3	3	3	5	1	1	1	4	3	4	5	4	52	23	7	12	5	5	1	3	4	1	200
6	Simple Simulation	4	1	4	3	4	4	4	3	3	2	4	1	4	1	4	1	4	3	4	58	19	8	18	8	5	1	3	4	1	500
7	Simple Linear CAT	3	1	1	3	3	1	2	3	3	2	3	1	2	1	3	1	3	3	3	42	15	6	10	7	4	1	3	4	1	100
8	Simple Linear CBT	3	1	2	3	3	2	2	3	1	2	3	1	2	1	3	1	3	3	3	42	15	6	10	7	4	1	3	4	1	435
9	Linear Mainframe CBT	3	1	2	4	1	2	1	3	1	2	3	1	3	1	3	1	3	3	3	41	15	7	9	6	4	1	3	4	1	80
10	Linear Mainframe CBT	2	1	1	4	1	2	1	3	1	2	2	1	3	1	2	1	2	3	4	37	11	7	8	6	5	1	3	4	1	220
11	Simple CBT	3	3	3	3	2	3	2	3	3	2	3	1	3	1	3	3	3	3	3	50	17	9	14	6	4	1	3	4	1	180
12	Simple CBT	3	3	3	3	2	3	2	3	3	2	3	1	3	1	3	3	3	3	2	49	17	9	14	6	3	1	3	4	1	250
13	Simple CBT	4	3	3	3	2	3	2	3	3	2	4	1	3	1	4	3	4	3	3	54	21	9	14	6	4	1	3	4	1	100
14	Simple CBT	3	3	3	3	2	3	2	3	3	2	3	1	3	1	3	3	3	3	3	50	17	9	14	6	4	1	3	4	1	320
15	TNA Project	2	3	3	2	3	2	3	3	3	1	2	1	3	1	2	3	2	5	1	45	12	11	14	6	2	5	2	4	3	78
16	Managers and the Law	1	3	4	1	3	3	4	4	3	2	1	1	3	2	1	1	1	4	2	44	7	10	18	6	3	5	2	5	0.75	420
17	Remedial Maths	3	1	2	4	2	3	3	3	1	3	2	1	3	1	3	3	3	3	5	49	17	7	12	7	6	5	4	4	3	1150
18	Oscilloscope Simulation	3	5	4	4	3	4	4	3	3	2	3	1	3	1	3	1	3	3	5	58	15	11	18	8	6	5	4	4	5.2	1370
19	Sine Ratio Concept	3	3	2	4	2	3	3	3	1	1	3	1	3	1	3	1	3	3	5	48	14	9	12	7	6	5	4	4	1.2	324
20	Multimeter Simulation	3	3	3	4	3	4	4	3	5	3	3	1	3	1	3	3	3	4	5	61	18	10	19	8	6	5	4	4	4	2238
21	Learner Control	3	3	2	3	2	3	3	3	3	3	3	1	3	1	3	3	3	3	3	51	18	9	14	6	4	5	4	4	2	1429
22	Recipe Conversion	3	1	2	3	1	3	2	3	3	3	3	1	2	1	3	3	3	3	3	46	18	6	13	5	4	5	4	4	8.8	680
23	Oscilloscope Guidance	4	5	3	4	3	4	4	3	3	4	4	1	3	1	4	3	4	4	5	66	23	12	17	8	6	5	4	3	12	3700
24	Part Task Trainer	3	3	4	4	2	4	3	3	3	4	3	1	3	1	3	3	3	4	5	59	19	10	17	7	6	5	4	4	3	1441
25	Electricity Terms	5	1	2	3	1	4	1	3	1	1	5	1	2	1	5	5	5	3	1	50	26	6	11	5	2	5	4	4	1.64	185.51
26	Safe Use of Electicity	4	1	2	3	1	4	1	3	1	1	4	1	2	1	4	5	4	3	1	46	22	6	11	5	2	5	4	5	0.62	88
27	Household Circuit Game	1	1	3	3	1	4	1	3	3	1	1	1	2	1	1	1	1	3	1	33	6	6	14	5	2	5	4	5	0.8	135.59
28	Cost of Using Electricity	2	1	2	3	1	4	1	3	1	1	2	1	2	1	2	3	2	3	1	36	12	6	11	5	2	5	4	5	0.39	32.3
29	WinEcon 1 (Betal 3.3)	3	1	2	3	2	3	3	3	1	2	3	1	2	1	3	3	3	4	5	48	17	7	12	6	6	5	2	1	0.6	50
30	WinEcon 2 (Betal 3.3)	3	3	2	3	2	3	3	3	1	2	4	1	3	1	4	5	3	4	4	54	21	10	12	6	5	5	2	1	3.4	96
31	WinEcon 3 (Betal 3.3)	5	3	2	3	2	3	3	3	1	2	5	1	3	1	5	5	4	4	5	60	26	10	12	6	6	5	2	1	2.1	100
32	WinEcon 4 (Betal 3.3)	5	3	2	3	2	3	3	3	1	2	5	1	3	1	5	5	5	4	5	61	27	10	12	6	6	5	2	1	4.7	480
33	WinEcon 5 (Betal 3.3)	5	3	2	3	2	3	3	3	1	2	5	1	3	1	5	5	5	4	5	61	27	10	12	6	6	5	2	1	2.5	175
34	WinEcon 6 (Betal 3.3)	4	5	3	3	2	3	3	3	1	2	4	1	2	1	4	3	4	4	5	57	21	11	13	6	6	5	2	1	3.1	175
35	WinEcon 7 (Betal 3.3)	5	3	2	3	2	3	3	3	1	2	5	1	2	1	5	3	5	4	5	58	25	9	12	6	6	5	2	1	2.6	300
36	WinEcon 8 (Betal 3.3)	3	1	2	3	2	3	3	3	1	2	3	1	2	1	3	3	3	4	5	48	17	7	12	6	6	5	2	1	2.9	270
37	WinEcon 10 (Betal 3.3)	5	1	2	3	2	3	3	3	1	2	5	1	2	1	5	3	5	4	5	56	25	7	12	6	6	5	2	1	2.1	150
38	WinEcon 11 (Betal 3.3)	5	1	2	3	2	3	3	3	1	2	5	1	2	1	5	3	5	4	5	56	25	7	12	6	6	5	2	1	2.6	240
39	WinEcon 12 (Betal 3.3)	5	1	1	3	2	3	3	3	1	2	5	1	2	1	5	3	5	4	5	55	25	7	11	6	6	5	2	1	3.5	210
40	WinEcon 13 (Betal 3.3)	4	3	1	3	2	3	3	3	1	2	4	1	2	1	4	3	4	4	5	53	21	9	11	6	6	5	2	1	4.3	210
41	WinEcon 14 (Betal 3.3)	3	3	1	3	2	3	3	3	1	2	3	1	2	1	3	3	3	4	5	49	17	9	11	6	6	5	2	1	4.4	210
42	WinEcon 15 (Betal 3.3)	3	3	1	3	2	3	3	3	1	2	3	1	2	1	3	3	3	4	5	49	17	9	11	6	6	5	2	1	2.8	200
43	WinEcon 16 (Betal 3.3)	3	3	2	3	2	3	3	3	1	2	3	1	2	1	3	3	3	4	5	50	17	9	12	6	6	5	2	1	3.3	180
44	WinEcon 17 (Betal 3.3)	2	3	2	3	2	3	3	3	1	2	2	1	2	1	2	3	2	4	5	46	13	9	12	6	6	5	2	1	1.5	200
45	WinEcon 18 (Betal 3.3)	2	3	2	3	2	3	3	3	1	2	2	1	2	1	2	3	2	4	5	46	13	9	12	6	6	5	2	1	3.4	312
46	WinEcon 19 (Betal 3.3)	5	1	1	3	2	3	3	3	1	2	5	1	2	1	5	3	5	4	5	55	25	7	11	6	6	5	2	1	3.6	300
47	WinEcon 20 (Betal 3.3)	1	1	1	3	2	3	3	3	1	2	1	1	2	1	1	3	1	4	5	39	9	7	11	6	6	5	2	1	1.3	175
48	WinEcon 21 (Betal 3.3)	3	1	1	3	2	3	3	3	1	2	3	1	2	1	3	3	3	4	5	47	17	7	11	6	6	5	2	1	2.3	60
49	WinEcon 22 (Betal 3.3)	5	3	1	3	2	3	3	3	1	2	5	1	2	1	5	3	5	4	5	57	25	9	11	6	6	5	2	1	4	450
50	WinEcon 23 (Betal 3.3)	3	3	1	3	2	3	3	3	1	2	5	1	2	1	3	3	3	4	5	51	19	9	11	6	6	5	2	1	2.1	100
51	WinEcon 24 (Betal 3.3)	3	3	1	3	2	3	3	3	1	2	5	1	2	1	3	3	3	4	5	51	19	9	11	6	6	5	2	1	2.9	150
52	WinEcon 25 (Betal 3.3)	3	3	1	3	2	3	3	3	1	2	3	1	3	1	3	3	3	4	5	50	17	10	11	6	6	5	2	1	1.5	150
53	WinEcon 26 (Betal 3.3)	2	3	1	3	2	3	3	3	1	2	2	1	3	1	2	3	2	4	5	46	13	10	11	6	6	5	2	1	1.8	150
54	PCCAL	5	5	3	2	3	4	4	3	3	1	1	1	3	3	5	5	5	4	5	65	22	12	17	8	6	5	2	1	6	600
55	Interactive Video	1	3	3	2	3	4	4	3	3	1	1	1	3	5	1	5	1	4	3	51	10	10	17	10	4	5	2	4	10	2572
56	Effectiveness	5	3	3	3	3	4	4	3	3	1	1	1	3	1	5	5	1	3	1	53	18	9	17	7	2	1	3	5	0.42	177
57	Situational Leadership	5	3	3	3	3	4	4	3	3	1	1	1	3	1	5	5	1	3	1	53	18	9	17	7	2	5	2	5	0.63	531
58	Distributed Management in Education	5	3	3	3	3	4	4	3	3	1	1	1	3	1	5	5	1	3	1	53	18	9	17	7	2	5	2	5	0.53	323
59	Hoy-Miskal-Bridges	5	3	3	3	3	4	4	3	3	1	1	1	3	1	5	5	1	3	1	53	18	9	17	7	2	5	2	5	0.65	331
60	Lewin Forces	5	3	3	3	3	4	4	3	3	1	1	1	3	1	5	5	1	3	1	53	18	9	17	7	2	5	2	5	0.48	237
61	Simulation 1	1	5	4	2	5	4	5	3	5	3	1	1	5	3	1	1	1	3	5	58	8	13	21	10	6	1	2	4	1	1044
62	Simulation 2	1	3	3	2	4	4	4	3	3	1	1	1	4	2	1	1	1	3	5	49	8	10	17	8	6	1	2	4	1	166
63	Simulation 2	1	1	2	2	3	4	4	3	1	3	1	1	3	2	1	1	1	3	5	42	8	7	14	7	6	1	2	4	1	71
64	Simulation 2	1	3	3	2	3	4	4	4	1	3	1	1	3	2	1	1	1	3	5	46	8	9	16	7	6	1	2	4	1	220
65	Statistics	5	3	3	2	3	3	4	3	3	1	5	1	3	1	5	5	5	3	1	53	26	9	16	6	6	5	2	2	20	2500
66	HyperCard Tutorail	5	1	4	2	3	1	4	1	3	1	5	1	2	1	5	5	5	3	1	53	26	6	13	6	2	5	2	5	0.5	8
67	Radio-Telephone Tutorial (Linkway)	1	3	5	2	3	5	4	3	3	1	1	1	3	1	1	3	1	3	1	45	8	9	20	6	2	5	2	4	3	155

Table 7.2 - Library courseware effort log

Week	learning	specification	instructional design	multimedia design	multimedia developmer	integration	testing	other
1	35.5	0	0	0	0	0	0	1.5
2	0	6.5	0	0	0	0	0	14.5
3	0	4	0	30	0	0	0	0
4	6	2	2.5	21	3.5	0	0	0
5	27.5	2.5	0	9	2	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	4	12.5	0	0	0	0
8	5.5	5	0	13	0	0	0	0
9	0	2	6	14.5	16	0	0	0
10	0	0	12	3	16.5	0	10	0
11	0	0	0	0	14.5	0	0	0
12	0	2.5	0	7.5	0	2	1	0
	74.5	24.5	24.5	110.5	52.5	2	11	16

Table 7.3 - Student data form library courseware

Student	Learner time	Pre-test	Post-test	MGS
1	20	1	5	0.8
2	30	2	4	0.5
3	26	1	4	0.75
4	28	2	7	0.71
5	28	3	3	0
6	24	2	7	0.71
7	23	4	8	0.5
8	22	2	3	0.33
9	26	1	6	0.83
10	27	0	3	1
11	26	0	6	1
12	25	1	7	0.86
13	25	2	4	0.5
14	24	3	7	0.57
15	24	2	7	0.71
16	24	3	9	0.67
17	29	4	7	0.43
18	25	1	7	0.86
19	22	1	6	0.83
20	23	2	8	0.75
21	26	2	7	0.71
22	24	2	6	0.67
23	27	0	3	1
24	26	4	5	0.2
25	24	5	6	0.17

Table 7.4 - Library courseware code-based measures

Page	Media Objects	Overlays	Interactions	SLOC
1	1			11
2				
3				7
4	1			3
5	1			12
6				9
7				12
8	1			6
9				11
10				6
11				6
12				9
13				6
14				12
15				13
16	1			25
17	1			40
18				11
19	1			28
20				25
21	1			30
22			1	11
23			1	34
24				
25	1		1	29
26			1	25
27				6
28				
29	1			17
30	4			22
31				3
32	1			3
33				3
34				3
35	2			12
36				3
37				3
38	1			3
39	1			60
40	1			51
41	1			75
42	1			48
43	1			52
44				24
45				18
46				18
47				15
48				6
49				9
50				6
51				12
52				6
53				6
54				9
55				9
56				6
57				6
58				
59				
60				9
61				9
62				9
63				6
64				6
65				12
66				6
67	1			3
68	1			3

Table 7.5 - Household electricity module one code-based measures

Module	Page	Blank	Text	Coments	Executable	Physical	Screens	Interactions
M1	0	2	0	13	0	15	0	0
M1	1	0	22	1	14	34	3	0
M1	2	1	9	1	28	37	3	0
M1	3	0	4	0	36	37	3	0
M1	4	0	8	0	35	37	3	0
M1	5	0	5	0	33	37	1	0
M1	6	1	3	0	13	16	3	2
M1	6	1	13	0	10	21	5	2
M1	7	0	29	0	23	37	15	3
M1	8	0	30	0	20	37	14	8
M1	9	0	26	0	23	37	13	4
M1	10	0	32	0	16	37	12	5
M1	11	0	32	0	18	37	11	2
M1	12	0	30	0	20	37	12	4
M1	13	1	30	0	18	36	12	6
M1	14	0	12	0	32	37	12	6
M1	15	1	17	0	27	37	12	9
M1	16	3	6	0	11	19	3	2
M1	16	0	14	0	7	18	4	2
M1	17	0	29	0	21	37	14	6
M1	18	0	22	0	27	37	13	8
M1	19	1	28	0	18	37	11	6
M1	20	0	12	0	34	37	11	9
M1	21	1	14	0	28	37	10	1
M1	22	2	8	0	7	17	1	3
M1	22	0	12	0	13	20	6	3
M1	23	0	32	0	14	37	8	5
M1	24	0	29	0	21	37	13	7
M1	25	0	30	0	20	37	13	7
M1	26	1	23	0	18	37	10	4
M1	27	0	17	0	28	37	9	6
M1	28	1	20	0	25	37	10	2
M1	29	2	23	0	14	37	4	3
M1	30	0	29	0	20	37	12	7
M1	31	0	33	0	12	37	8	4
M1	32	0	29	0	21	37	13	8
M1	33	1	11	0	8	14	6	1
M1	33	0	12	0	14	22	4	3
M1	34	0	23	0	25	37	10	6
M1	35	1	18	0	30	37	12	9
M1	36	0	19	0	25	37	9	0
M1	37	1	19	0	14	32	4	0
		21	814	15	841	1374	352	163

Table 7.6 - Household electricity module two code-based measures

Module	Page	Blank	Text	Coments	Executable	Physical	Screens	Interactions
M2	37	0	0	0	2	5	0	0
M2	38	0	21	0	20	37	9	4
M2	39	0	25	0	19	37	8	6
M2	40	0	31	0	18	37	12	5
M2	41	0	28	0	21	37	12	8
M2	42	0	32	0	16	37	11	5
M2	43	0	31	0	20	37	12	6
M2	44	1	17	0	31	37	11	5
M2	45	0	26	0	23	37	12	4
M2	46	1	17	0	28	37	9	5
M2	47	0	28	0	21	37	14	7
M2	48	0	31	0	18	37	12	3
M2	49	0	30	0	16	37	9	6
M2	50	1	17	0	30	37	12	5
M2	51	0	22	0	27	37	12	4
M2	52	1	17	0	28	37	10	5
M2	53	0	30	0	21	37	14	7
M2	54	0	29	0	17	37	9	6
M2	55	0	31	0	18	37	12	6
M2	56	1	20	0	22	37	10	4
M2	57	0	19	0	26	37	12	3
M2	58	1	15	0	26	37	7	4
M2	59	0	27	0	17	37	8	8
M2	60	0	5	0	30	37	3	3
M2	61	1	1	0	1	3	0	0
		7	550	0	516	859	240	119

Table 7.7 - Household electricity module three code-based measures

Module	Page	Blank	Text	Coments	Executable	Physical	Screens	Interactions
M3	61	0	12	0	22	34	7	3
M3	62	1	30	0	12	37	7	3
M3	63	1	24	0	19	37	9	0
M3	64	1	10	0	31	37	4	8
M3	65	2	26	0	14	37	6	3
M3	66	2	20	0	18	37	4	6
M3	67	1	20	0	14	37	3	3
M3	68	2	27	0	9	37	3	0
M3	69	2	29	0	5	37	2	0
M3	70	1	21	0	26	37	12	5
M3	71	1	17	0	26	37	11	4
M3	72	2	19	0	20	37	7	4
M3	73	0	20	0	27	37	10	8
M3	74	2	22	0	17	37	6	3
M3	75	2	18	0	25	37	9	3
M3	76	0	1	0	37	37	1	14
M3	77	0	0	0	37	37	0	19
M3	78	0	0	0	37	37	0	18
M3	79	0	1	0	37	37	1	18
M3	80	0	0	0	37	37	0	18
M3	81	0	1	0	37	37	1	19
M3	82	1	9	0	30	37	4	4
M3	83	0	21	0	27	37	11	11
M3	84	0	25	0	25	37	13	12
M3	85	0	25	0	24	37	12	12
M3	86	0	24	0	25	37	12	13
M3	87	0	23	0	26	37	12	12
M3	88	0	25	0	25	37	13	12
M3	89	0	25	0	24	37	12	12
M3	90	0	23	0	26	37	12	13
M3	91	0	25	0	25	37	13	9
M3	92	2	16	0	20	32	7	0

Table 7.8 - Household electricity module four code-based measures

Module	Page	Blank	Text	Coments	Executable	Physical	Screens	Interactions
M4	92	0	0	0	2	5	0	0
M4	93	0	25	0	17	37	7	5
M4	94	1	29	0	16	37	8	5
M4	95	0	31	0	15	37	11	4
M4	96	0	32	0	14	37	9	4
M4	97	1	26	0	20	37	10	4
M4	98	0	31	0	14	37	7	4
M4	99	0	32	0	18	37	12	5
M4	100	0	32	0	18	37	13	4
M4	101	0	34	0	12	37	9	3
M4	102	0	32	0	14	37	7	5
M4	103	0	26	0	20	37	12	4
M4	104	4	17	0	17	37	5	3
M4	105	0	0	0	0	1	0	0
E	106	2	13	0	0	15	0	0
		8	360	0	197	465	110	50

Table 7.9 - WinEcon Code-based data set

chapter	page_count	user_properties	objects	graphics	fields	static_fields	edit_fields	listbox_fields	scroll_fields	words	characters	buttons	push_buttons	radio_buttons	check_buttons	graphic_buttons	captioned_buttons	combo_boxes	groups	level_one_objects	scripted_objects	propertied_objects	named_objects	nothy_objects	WinEcon_type	WinEcon_text_cards	WinEcon_prof_fields	WinEcon_popups	WinEcon_graphs	WinEcon_spreadshee	raw_lines	blank_lines	comment_lines	code_lines
1	6	22	903	43	398	398	0	0	3	6861	39867	92	45	10	0	61	32	0	98	149	7	205	377	0	*	8	6	8	2	0	2294	358	156	1780
2	34	36	5666	115	2125	2115	10	0	0	14342	82132	498	208	97	22	367	183	12	451	1739	331	1400	1865	0	*	29	24	81	29	0	13501	1465	821	11215
3	21	21	3166	84	1012	1000	12	1	5	6475	34987	236	151	8	17	206	98	0	390	799	160	866	906	0	*	13	15	76	18	0	3804	124	15	3665
4	47	162	6127	248	2437	2357	80	3	0	12902	72409	310	119	4	0	267	79	0	915	1713	224	1391	2559	24	*	45	28	59	11	0	11663	815	689	10159
5	25	29	2661	109	845	840	5	1	2	4545	27066	322	142	72	3	213	140	0	385	805	242	956	975	0	*	15	21	61	5	0	4471	352	30	4089
6	31	53	2572	139	883	883	0	0	0	7761	47258	364	232	39	2	239	106	0	408	812	228	862	928	0	*	19	31	61	0	0	3900	233	8	3659
7	26	69	2630	142	888	888	0	0	5	6417	38786	342	165	73	4	229	134	0	399	949	208	1012	859	0	*	19	24	103	7	0	4099	221	39	3839
8	29	71	3871	156	1564	1529	35	1	0	12602	71334	153	44	0	0	148	33	0	559	1532	136	941	1544	39	*	27	25	19	14	0	9951	444	132	9375
10	21	46	2468	110	869	866	3	10	5	7316	42770	202	75	34	9	138	84	7	297	621	43	733	700	5	*	21	19	17	19	0	5826	1081	284	4461
11	26	58	2698	105	850	839	11	0	0	8547	50706	223	43	33	6	174	45	0	315	1107	194	1002	1383	0	*	24	9	16	3	0	8070	1273	573	6224
12	35	105	2936	209	1157	1139	18	0	0	11134	66333	282	121	27	0	219	93	0	391	1026	135	861	1207	7	*	35	26	50	1	0	8026	2018	1013	4995
13	43	80	3073	228	1249	1249	0	0	0	24549	148528	270	106	17	0	215	71	0	520	818	127	1005	1172	4	*	41	21	59	0	0	5664	1184	854	3626
14	44	87	3293	226	1292	1292	0	1	0	18881	111814	261	89	14	0	228	61	0	497	1138	124	1150	1519	4	*	43	27	45	0	0	7816	1776	906	5134
15	28	57	2735	156	1017	1017	0	1	0	12710	72224	193	72	5	4	161	52	0	392	1078	42	1274	1323	0	*	31	19	40	0	0	7150	809	28	6313
16	33	70	4530	137	1674	1496	178	0	0	15511	90386	323	146	74	0	224	119	0	319	2052	278	1192	2394	0	*	27	9	59	15	0	12677	1616	735	10326
17	15	25	2654	86	1096	1096	0	0	0	6752	36814	172	90	9	6	102	81	0	357	592	18	757	581	0	*	19	8	19	41	0	6779	1418	511	4850
18	34	41	3625	120	805	782	23	2	12	8697	57254	328	91	39	35	190	129	1	384	1004	330	739	1573	0	*	20	16	17	3	0	8323	1402	658	6263
19	36	69	4120	204	1389	1379	10	1	13	9304	53351	484	188	99	43	303	197	0	612	1242	228	1589	1291	0	*	37	28	107	18	0	5473	329	47	5097
20	13	23	1163	65	401	401	0	0	0	3487	20173	128	82	6	1	120	39	0	197	260	101	386	326	0	*	7	15	46	3	0	1698	104	7	1587
21	23	24	1497	78	587	578	9	19	1	8590	56425	258	100	29	2	149	133	0	204	721	145	537	750	0	*	18	14	6	1	0	4469	651	135	3683
22	40	136	3396	272	1359	1340	19	0	1	17573	112022	444	144	51	0	350	166	3	687	1093	161	1051	1781	5	*	43	32	43	2	0	9008	673	241	8094
23	21	23	2355	68	927	902	25	0	0	6902	36735	292	207	36	0	116	159	0	405	537	279	687	797	0	*	9	21	72	6	0	3374	170	2	3202
24	29	52	5158	210	2477	2471	6	2	0	10999	56537	386	192	44	29	288	137	0	782	1129	234	1189	1437	0	*	17	39	112	15	0	4919	162	16	4741
25	15	39	2327	69	509	485	24	0	0	1651	8446	97	69	5	0	77	65	0	431	682	117	423	436	2	*	2	13	2	11	0	4022	710	42	3270
26	18	39	2548	100	774	773	1	0	0	3663	20104	134	98	12	0	97	67	0	376	745	193	591	579	2	*	2	15	37	19	0	6369	871	303	5195

Table 7.10 - WinEcon Code-based data set

ifs	elses	conditions	whens	steps	whiles	dos	sends	to_gets	to_sets	to_handles	notify_handlers	forwards	breaks	continues	system_variables	local_variables	trailing_comments	pagescript_raw_lines	development_time (hours)	elapsed time (days)	Full_time_equivalent_team	Total_team_size (Physical Bodies)	Effort (developer hours)	to_get+to_set (C44+C45)	pagescript_raw_lines/raw_lines (C54/C32)	properties_objects/named_objects (C23/C24	scripted_objects/pagescrip_raw_lines (C22/C	technical_quality	content_quality	start_programmer	mid_programmer	end_programmer	page_count	Institution
183	60	15	62	35	0	1	164	2	2	66	0	20	23	2	42	139	56	2262	300	40	1.25	1.5	50	4	0.986051	0.54377	0.003095	3	5	19	3	3	6	7
474	82	222	922	198	3	6	983	3	0	655	0	349	40	0	429	189	125	10163	450	60	1.6	2	96	3	0.752759	0.75067	0.032569	4	5	19	9	9	34	8
149	46	23	81	99	2	10	499	2	0	229	0	177	30	3	312	175	58	1964	600	80	1.25	2	100	2	0.516299	0.95585	0.081466	2	5	19	7	7	21	3
416	172	143	585	159	0	4	969	30	0	471	24	276	67	1	128	150	44	6811	2400	320	1.5	1.7	480	30	0.583984	0.54357	0.032888	2	5	13	13	8	47	5
95	21	37	156	41	0	4	652	0	0	294	0	256	6	3	196	88	48	2765	*	140	1.25	*	175	0	0.61843	0.98051	0.087523	2	1	12	2	9	25	1
65	2	44	154	25	0	2	651	0	0	283	0	234	4	9	54	32	6	2515	900	140	1.25	2	175	0	0.644872	0.92888	0.090656	2	1	12	2	9	31	1
186	45	28	99	86	1	6	528	0	0	267	0	225	33	0	298	122	90	2146	1950	240	1.25	2	300	0	0.523542	1.17811	0.096925	3	5	15	7	7	26	3
477	185	139	721	136	6	3	736	3	0	315	39	212	84	2	101	164	50	6601	1350	180	1.5	1.3	270	3	0.66335	0.60946	0.020603	4	5	13	13	8	29	5
249	83	66	240	173	0	0	549	4	0	171	5	75	36	3	32	446	284	5545	900	120	1.25	1.6	150	4	0.951768	1.04714	0.007755	3	3	3	3	3	21	7
343	41	88	455	131	0	7	835	8	0	348	0	180	4	0	71	91	77	7084	1200	160	1.5	1	240	8	0.877819	0.72451	0.027386	3	3	9	9	9	26	8
103	31	69	345	90	0	5	706	2	0	229	13	100	0	12	54	150	47	6417	997	140	1.5	2	210	2	0.799527	0.71334	0.021038	1	5	17	16	3	35	2
82	30	54	237	5	0	0	487	0	0	166	9	83	4	8	28	167	5	3897	997	140	1.5	1.7	210	0	0.68803	0.85751	0.032589	1	5	16	16	3	43	2
96	38	77	355	13	0	0	688	0	0	173	11	72	0	9	56	70	10	6192	997	140	1.5	1.7	210	0	0.792221	0.75708	0.020026	1	5	16	16	3	44	2
283	30	88	392	285	0	1	1302	2	0	171	0	67	7	23	108	173	9	7066	1200	160	1.25	2	200	2	0.968252	0.96296	0.005944	3	4	3	3	3	28	7
408	13	123	523	166	0	3	1376	38	0	501	0	236	49	1	366	297	17	10822	900	120	1.5	1	180	38	0.853672	0.49791	0.025688	3	4	9	9	9	33	8
238	100	46	134	106	2	2	673	7	0	164	0	43	46	7	225	479	296	6767	1200	160	1.25	2	200	7	0.99823	1.30293	0.00266	4	4	3	3	3	15	7
248	108	64	236	122	0	4	827	36	0	720	0	385	2	1	82	390	51	5102	1915	240	1.3	1.3	312	36	0.613	0.4698	0.064681	1	5	5	5	9	34	8
298	80	46	215	115	1	8	580	2	0	331	0	251	18	8	323	115	174	3389	1915	240	1.25	2	300	2	0.619222	1.23083	0.067276	1	5	7	7	7	36	3
30	8	24	71	5	0	1	436	0	0	125	0	115	0	4	33	17	3	797	1350	140	1.25	2	175	0	0.469376	1.18405	0.126725	2	1	12	2	7	13	1
213	72	53	255	53	1	5	355	17	1	297	0	129	14	0	285	131	11	3453	*	40	1.5	*	60	18	0.772656	0.716	0.041992	2	5	5	5	9	23	8
388	110	121	679	182	7	3	664	4	0	368	5	250	17	2	266	205	93	6910	2250	300	1.5	1	450	4	0.767096	0.59012	0.0233	1	3	13	14	8	40	5
75	36	19	99	93	0	1	498	0	0	312	0	281	1	3	5	39	3	1205	1350	80	1.25	2	100	0	0.357143	0.86198	0.231535	4	4	12	12	7	21	1
160	29	42	232	128	1	9	621	0	0	276	0	241	15	1	205	141	126	2317	975	120	1.25	2	150	0	0.471031	0.82742	0.100993	2	4	7	7	7	29	3
118	36	17	80	150	5	0	387	31	0	250	2	89	8	1	297	17	0	2388	*	120	1.25	*	150	31	0.593734	0.97018	0.048995	1	5	4	4	8	15	6
247	104	30	122	199	5	5	322	49	7	426	4	210	29	10	359	20	3	3022	*	120	1.25	*	150	56	0.474486	1.02073	0.063865	2	5	4	4	8	18	6

Table 7.11 - WinEcon object-based data set

Media Objects	Screen Overlays	Screens Total	Interactions	Objectives	Learner tme (Lh
7	50	56	5	1	0.6
46	220	254	17	6	3.4
38	92	113	7	8	2.1
51	251	298	13	7	4.7
32	120	145	10	6	2.5
19	146	177	10	4	3.1
23	110	136	13	9	2.6
39	356	385	26	7	2.9
19	100	121	6	4	2.1
20	161	187	24	6	2.6
23	199	234	34	7	3.5
27	289	332	50	7	4.3
30	288	332	6	6	4.4
26	182	210	27	5	2.8
30	231	264	45	6	3.3
24	81	96	6	3	1.5
59	173	207	30	6	3.4
24	173	209	12	8	3.6
8	39	52	3	6	1.3
15	147	170	12	6	2.3
12	296	336	27	6	4
6	124	145	20	4	2.1
13	170	199	46	9	2.9
14	52	67	2	5	1.5
15	63	81	5	8	1.8
620	4113	4806	456	150	69.3

7.4 REFERENCES

Casey, R.J. et al., 1988. Capturing skill in estimating training development effort: A follow-up final report. *Performance & Instruction*. **27(9)**: pp. 40-44.

Lee, C. and Zemke, R., 1987. How long does it take? *Training*. **24(6)**: pp. 75-80.

Mikos, R. et al., 1987. Estimating training effort/cost: A follow-up 'final report' (Part 1). *Performance & Instruction*. **26(5)**: pp. 24-29.