Journal of Universal Computer Science, vol. 4, no. 3 (1998), 308-325
submitted: 10/12/97, accepted: 28/12/97, appeared: 28/3/98

Estimation Metrics for Courseware Maintenance Effort

Christine W. Thackaberry
(Richland, Washington
tberrys@owt.com)

Roy Rada
(Washington State University, Washington
rada@eecs.wsu.edu)

Abstract: Software engineering methods and metrics to estimate development time for the development and maintenance of computer-based training (CBT) differ from methods and metrics used to develop large information systems. The estimation techniques for large information systems employ Lines-Of-Code and Feature/Function points to calculate project effort in staff-months [Boehm 1981]; techniques that are difficult to apply to CBT effort estimation. For the development of computer-based training systems, Development to Delivery Time Ratios have been the usual estimation metric, but these also have limitations [Marshall et al. 1995]. Metrics to accurately measure the development effort of Multimedia Information Systems (MIS) are currently being developed and investigated [Fletcher et al. 1997], but still differ from computer-based training systems development. This paper presents an estimation model for effort development of small courseware projects (less than 2 staff-months). By identifying the sub-tasks of the development phase, an individual estimation technique is suggested for each sub-task. The sum of all sub-tasks estimations determines the total effort estimation for the development phase of a particular lesson. Incorporating historical data as a baseline and identifying risk cost factors, this method is accurate for estimating effort of some sub-tasks and for the lesson unit as a whole. This method is not meant to be a „silver bullet“ [Brooks 1995] but a start toward building an accurate estimation tool and a refinement of the development process.
Categories: Software, Software Engineering

1 Introduction.

Software estimation metrics and use of historical development data are important aspects of all types of software development. The capacity to estimate software costs provides two primary keys for improving software productivity: a sounder baseline for planning and the control of software projects [Boehm 1981] [Stark et al. 1994]. The collection of development data provides the basis for software process adjustment and the avoidance of many development practices that lead to project failure [Jones 1996] [Brooks 1995]. The software project planner must estimate three things before a project begins:

- how long it will take,
- how much effort will be required, and
- how many people will be involved.

The measurements of software systems, function/feature points, or Lines-of-Code (LOC) metrics are designed to be effective for business, control, scientific, and system software [Pressman 1992] [Johnson 1991] [Symons 1991]. A computer-based training (CBT) development team does not and cannot measure its final product solely on the number of Delivered LOC. To meet strict deadline schedules and the need for a consistent presentation, computer-based training system developers implement time saving methods, such as use of authoring software, libraries of reusable code, and creation of reusable presentation sub-modules (or lesson templates) [Pizzini et al. 1997] [Veljkov 1990]. The final courseware product consists not only of a final software product, but also of lesson plans, and resources (graphics, audio, animations, and video) created to meet specific training requirements.

Typically, courseware estimation techniques are based on development to delivery time ratios. Delivery time refers to the amount of time a learner spends with the program. These estimations range from 25 to 400 staff-hours of development time for 1 hour of delivery time. Such estimation methods have significant limitations and weaknesses [Marshall et al. 1995]. For the courseware system discussed in this paper, delivery time varied from 45 to 420 minutes for a sample group of 886 learners. With such a wide range of delivery times one can not expect to reliably predict development time as a function of delivery time. Instead, in this paper the estimation of development times for courseware products will be based on analysis of the steps involved in developing courseware, the number of pages of courseware that are to be produced, and other aspects of the courseware development cycle. Estimation metrics will be presented along with supporting historical development data.

Interactive multimedia is one of the technologies that is most influencing the educating and training of people [Rada 1997]. To fulfill a company's training requirements in a cost-effective manner, company managers are exploring ways to replace current training with computer-based training. Consistency of presentation, decreased training time and costs, the ability for self-paced instruction, and student interactivity have increased the demand for this training.

After the development of the initial CBT systems, some enter into a constant evaluation/redesign phase. In systems developed to supply mandated annual safety training to government agencies, the redesign incorporates not only the need to satisfy changing regulations, but also the desire to present this information in a new, instructionally sound format. *It is important that estimation methods be developed to assist in predicting project efforts, not only for the creation of new CBT systems, but also for the redesign of current systems.*

This paper addresses the amended software engineering development process used in the redesign of a CBT system. Using historical data, estimation metrics are suggested that may be employed to estimate development effort for the redesign of CBT systems. It must be understood that given that the set of data is dependent on the expertise of individuals, and the type of authoring tools used, these metrics may have a limited application.

2 Framework

A framework, both definitions and courseware system concepts, is important in understanding the capabilities and limitations of an estimation technique. For this reason, the courseware system, its development process, and how it differs from information systems are presented in this section.

2.1 Courseware Description

Courseware is software-supported educational content offered in a pedagogically sound way to students. Standards for courseware are not widely adopted. Courseware prepared in one toolset is typically not compatible with courseware produced from a different toolset. Attempts to analyze the effectiveness of courseware or courseware development are inhibited by the lack of agreement on a language for talking about courseware. To address these problems, the aviation industry, which heavily invests in courseware, has produced some standards [AICC 1997] and increasing effort is being invested in standardization [Rada and Schoening 1997].

In this paper, courseware is defined as a system of presenting several groups of lessons using several submenu systems. The system contains entry and exit capability, along with learner and test data saving capabilities. A lesson is defined specific to the authoring software used in the development of this courseware. Since the historical data was gathered during the development using Asymetrix Multimedia Toolbook, a lesson is defined as a book consisting of pages of presentation material.

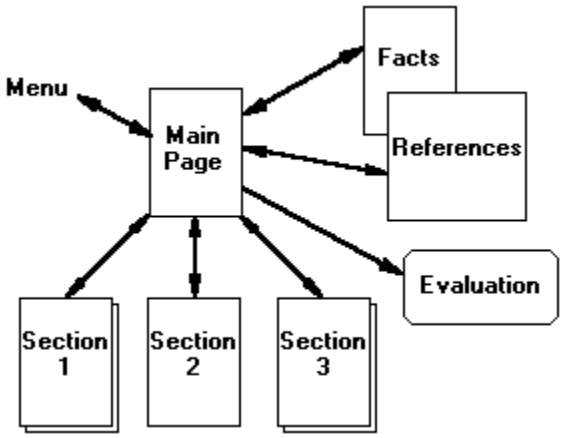

Figure 1: Standard Lesson.

A standard lesson [Figure 1] consists of these components:

- a main page, that contains an introduction to the lesson and all navigation controls,
- a facts page, that contains the summary of the content of the lesson,

- reference page, that contains the reasons the learner must be familiar with the contents of the lesson,
- several sections of content (each section possibly containing several pages), and
- an associated set of questions used to evaluate the learner's comprehension.

A page consists of a combination of text, graphics, audio files, video, and hypertext to present information to the learner. In the context of this paper, a lesson is defined as having 6 pages: main page, facts, references, three sections of one page each, and an evaluation of the content of the lesson.

2.2 Software Engineering Principles

„Metrics analysis begins with insight into the workings of the software development and maintenance processes" [Stark et. al. 1994]. The life cycles of software engineering [Pressman 1992]: the waterfall model, the prototype model, the spiral model, and the Fourth Generation model, assume an end in the main development of a system. The life cycle for courseware development, suggested by Marshall, is an amended waterfall model [Marshall et al. 1995].

Both models suggest a linear development of the software product whereas many of the tasks, particularly in the development phase, can be performed concurrently. Marshall's model introduces the necessity to define the start and end points that the estimation metrics measures. The amended life-cycle model [Figure 2] is utilized by the company that supplied the historical data and reflects its current courseware development process. The estimation metrics introduced in this paper measure the effort needed to perform all tasks within the start and end points depicted in this model. Normal maintenance activities handle minor changes to the courseware due to error detection, detection of missing or incorrect resources, or minor change requests from the customer. The evaluation phase is apart from normal maintenance activities. During this phase the effectiveness of the training is determined from statistics gathered from questions administered during the course. The results of the evaluation may create the need to redesign the system as a whole or in part.

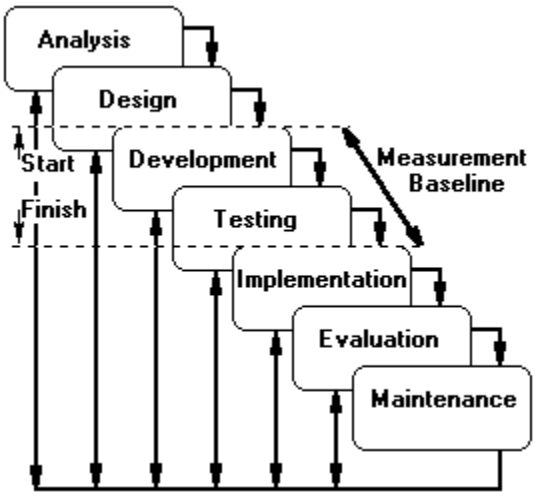

Figure 2: Amended courseware Life Cycle Model with start and finish points for the estimation model defined.

2.3 Organizational Planning

An information systems software development team consists of a chief programmer (systems engineer), technical staff, and a backup engineer [Pressman 1992]. The courseware development team [Figure 3] involves the concurrent efforts of instructional designers, technical editors, media specialists, programmers, and entry personnel [Rada 1995].

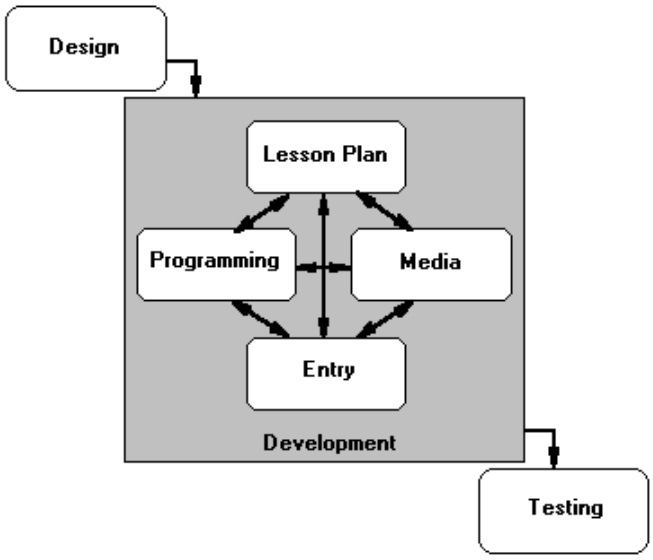

Figure 3: Development phase of Amended Life Cycle model.

The responsibilities of each team member are:

- instructional designer, who usually leads, plans, develops lesson plans, and coordinates all the activities of the team and sometimes acts as the technical editor for another instructional designer;
- technical editor, who verifies the clarity and consistency of the presented information and correct use of grammar and punctuation;
- resource specialist, who plans and coordinates the production of all graphics, audio, animation, and video;
- programmer, who plans and converts the requirement specifications into the courseware system, creates templates and utility programs, and maintains courseware documentation;
- entry personnel, who combines the lesson plans with the media to create the resulting lesson presentation; and
- testing personnel, who validates the accuracy of the entered content and media in an individual lesson.

Metrics for development effort estimations must be able to support predicting the effort of each sub-task of the development phase. These sub-tasks estimates are then incorporated into the main project plan by the project lead to calculate the total development effort.

3 Estimation Techniques

Some software project estimation techniques [Johnson 1990] [Boehm 1997] are based either on feature points or Lines-Of-Code. One of the computations in the

estimation process of feature points is based on the number of user inputs. Unlike a database retrieval system, a computer-based training course is created to be interactive, encouraging the learner to guide their own way throughout the course. Does each mouse click then count as one input (one feature point)? An attempt to calculate courseware effort with these methods was unsuccessful. A different base unit must be derived to calculate the effort involved in courseware development.

To determine effort estimations, a number of options are available [Pressman 1992]:

- delay estimation until late in the project,
- use decomposition techniques to generate project estimates,
- develop an empirical model, or
- acquire automated tools.

By subdividing the development phase and using historical data as a baseline value, estimation metrics can be developed to predict courseware development effort.

The use of authoring systems to assist in the development of courseware is common practice [Veljkov 1990], and a variety of authoring systems are readily available. The unit for information presentation in the authoring system utilized by the company that supplied the historical data is the page. A page consists of text, hypertext, and all associated media types. The estimation method presented will determine development effort by calculating effort associated with creating a single page.

3.1 Decomposition of The Development Phase

The development phase is divided into the following sub-tasks:

- Design(D_e): Creation of and technical editing of lesson plans. Total effort for the design (D_e) equals the sum of lesson creation (L_e) plus Technical editing (Te_e).
- Programming(P_e): Changes to the template as defined in the lesson plan, preliminary testing, and updates to all documentation.
- Media(M_e): Development of graphics, audio, video as defined in the lesson plan. Total effort for media development (M_e) equals the sum of graphic effort (G_e) plus audio effort (A_e) plus video effort (V_e).
- Entry(E_e): Input the lesson content into the template and coordinated the media presentation.
- Testing(T_e): Ensure the accuracy of the lesson presentation as an individual unit.

Total effort is the composite metric consisting of development effort for each sub-task:

$$\text{Total Effort} = D_e + P_e + M_e + E_e + T_e.$$

3.2 Use of Historical Data

Lack of historical measurement data often leads to setting arbitrary schedules for future projects, and is often the key to software failure [Jones 1996]. By incorporating historical data as a baseline in the estimation metrics presented in this paper, a solid empirical foundation is established.

The historical data collection employed a general reporting system, that is, time was reporting in the development of media, but was not subdivided by the media type. Discussions with the media staff and a review of their personal time logs were necessary to determine effort for each type of media (audio, graphics, and video). Discussions with the design personnel were also necessary to determine average time needed by the technical editor, since this time was also reported under the general heading of design.

The amount of effort reported for these sub-tasks varies from lesson to lesson as depicted in [Figure 4]. Reasons for these variances range from the experience of personnel, number and type of media developed, difficulty in the coordination of media types, and difficulty creating or obtaining media resources.

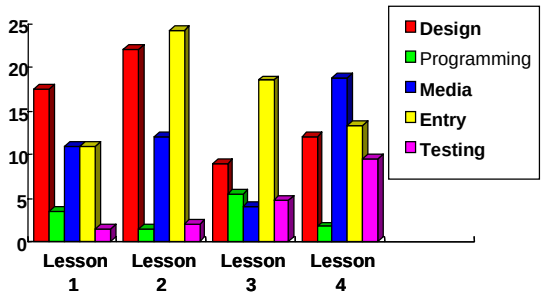

Figure 4: Reported Development Efforts (in staff-hours).

Although historical data was provided from a number of lessons, some of the data was discounted for the following reasons:

- inaccurate reporting due to the use of inappropriate reporting codes,
- implementation of a different reporting system in the middle of a lesson's development, or
- the lesson consisted of a group of sub-lessons, and therefore, did not conform to the definition of a standard lesson [Figure 1].

Sub-task	Average Staff Hours
Design Technical editing	2.25 / page + 4.0 * # of reviews
Audio Resource	0.25 / audio file
Graphic Resource	1.0 / graphic
Video Resource	4.0 per 20 seconds of video
Entry	1.25 / page
Programming	0.5 / page
Testing	0.25 / page

Table 2: Average page development time based on historical data.

By averaging the development effort reported, a baseline for calculating effort for each sub-task is introduced [Table 2]:

Design + Editing	$[D_e] = (2.25*N) + (4.0 * \#\text{ of reviews})$
Audio	$[A_e] = (0.25 *$ number of audio files$)$
Graphics	$[G_e] = (1.0 *$ number of graphics files$)$
Video	$[V_e] = (4.0 *$ number of video segments$)$
Entry	$[E_e] = (1.25 * N)$
Programming	$[P_e] = (0.5 * N)$
Testing	$[T_e] = (0.25 * N)$

Using these averages as baseline values, estimation metrics for each activity are developed as it relates to the development of one standard lesson.

Where N = number of pages in the lesson

$$\text{Total Effort} = D_e + A_e + G_e + V_e + E_e + P_e + T_e$$

For a 6-page lesson with one audio and graphics resource per page and two internal reviews, the estimation of effort is calculated as follows:

$D_e + A_e + G_e + V_e + E_e + P_e + T_e$

$D_e = [2.25 * 6] + [4.0 * 2] = 13.5 + 8.0 = 21.5$

$A_e = 0.25 * 6 = 1.5$ $G_e = 1.0 * 6 = 6.0$ $V_e = 4.0 * 0 = 0$

$E_e = 1.25 * 6 = 7.5$ $P_e = 0.5 * 6 = 3.0$ $T_e = 0.25 * 6 = 1.5$

Total Effort = 41.0 unadjusted staff-hours

This estimate does not include several tasks performed external to an individual lesson, such as incorporating this lesson into the course system, installation time, testing of the course as a whole after the addition of the lesson is complete, and a variety of other tasks that are not directly related to the development of a single lesson. It must also be noted that these metrics are also influenced by cost adjustment factors (discussed in the next section) and therefore, reflect unadjusted staff-hour estimates.

3.3 Cost Adjustment Factors

As with other software systems [Boehm 1981] [Pressman 1992], there are external constraints that have an influence on the actual effort in creating the final lesson, such as:

- the experience of each of the team members,
- availability of technical experts, information, and resources, and
- the existence or lack of templates.

Each activity associated with the development phase has its own constraints (cost adjustment factors) [Hackos 1996] that influence the estimation of development effort [Table 3].

Design [Cf_d]	DE * IA * SME * IR
Programming [Cf_p]	TA * PE * PC
Audio [Cf_a]	TE * RA
Graphics [Cf_g]	TE * RA
Video [Cf_v]	VE * RA
Entry [Cf_e]	EE * EC
Testing [Cf_t]	DL

DE: Design Experience
IA: Information Availability
SME: Subject Matter Expert(SME) availability
IR: Impact of Reviews on the design
TA: Template Adjustment
PE: Programming Experience
PC: Programming Complexity
TE: Tool Experience
RA: Resource Availability
VE: Video Experience
EE: Entry Experience
EC: Entry Complexity factor
DL: Detail Level of test

Table 3: Cost adjustment factors that influence a specific task.

The values [Hackos 1996] assigned to these cost adjustment factors are set within two ranges with 1.0 as the base value (reflecting no effect on the effort calculation). For adjustment factors that have a higher influence on actual effort [Table 4], the range is -

0.4 to 1.6 in increments of 0.1. For the remainder, the range is -0.5 to 1.5 in increments of 0.05. For example, an experienced designer would be given a design experience (DE) adjustment factor of 0.75, since we could expect the design task to be completed in less time than an average or less experienced designer. The design task is divided into two steps: lesson design and technical editing. The technical editing portion of the design task is not effected by the cost adjustment factors mentioned here.

Cost factor value	Cost Adjustment Factor
1.0 ± (0.1 to 0.6)	Resource Availability Entry Complexity Course Complexity Impact of Reviews on Design Template Adjustment
1.0 ± (0.05 to 0.5)	Design Experience Information Availability Subject Matter Expert(SME) availability Programming Experience Tool Experience Video production Experience Entry Experience Detail Level (of testing)

Table 4: Summary of Cost Factor Values

Calculating these cost factors, as it applies to the example introduced in Section 3.2:

Sub-task			Cost Factor Value
Design	$[Cf_d]$ =	DE * IA * SME * IR 1.1 * 1.0 * 1.0 * 1.2	= 1.32
Programming	$[Cf_p]$ =	TA * PE * PC 0.9 * 0.95 * 1.0	= 0.855
Audio	$[Cf_a]$ =	TE * RA 0.95 * 1.0	= 0.95
Graphics	$[Cf_g]$ =	TE * RA 0.95 * 1.1	= 1.045
Entry	$[Cf_e]$ =	EE * EC 1.0 * 1.2	= 1.2
Testing	$[Cf_t]$ =	DL 0.9	= 0.9

Applying these resulting cost factors to each sub-task effort for the example calculation:

sub-task		U Hrs*	Cost factor			C Hrs*
Design	(2.25 * 6) =	13.5	* 1.32 ≅	17.8	+ 8.0	25.8
Programming	(0.5 * 6) =	3.0	* 0.855 ≅	2.5		2.5
Audio	(0.5 * 6) =	1.5	* 0.95 ≅	1.4		1.4
Graphics	(1.0 * 6) =	6.0	* 1.045 ≅	6.2		6.2
Entry	(1.25 * 6) =	7.5	* 1.2 ≅	9.0		9.0
Testing	(.25 * 6) =	1.5	* 0.9 ≅	1.3		1.3
		33.0		38.2		46.2
* U Hrs=Unadjusted Hours C Hrs=Calculated Hours						

Estimated hours: 46.2.

Summary of estimation metrics for CBT development (in staff-hours):

Sub-task	
Design	(2.25 * N) * Cf_d + (4.0 * # reviews)
Audio	(0.25 * number of audio files) * Cf_a
Graphics	(1.0 * number of files) * Cf_g
Video	(4.0 * number of segments) * Cf_v
Entry	(1.25 * N) * Cf_e
Programming	(0.5 * N) * Cf_p
Testing	(0.25 * N) * Cf_t

Where N = number of pages

4 Results

The value of a software estimation model can be measured by the following criteria:

- comparison of actual effort hours to calculated hours,
- evaluation of the model against criteria used to measure the goodness of software cost estimation model [Boehm 1981].

The next two subsections present the results according to these two different criteria.

4.1 Comparison of Actual vs. Calculated Hours

The results of the estimation method presented in this paper are contained in [Table 5]. The reported hours represent actual production hours from historical data, while calculated hours represent the result of applying cost factors to initial results of the estimation metrics (unadjusted hours) introduced in [Table 2]. The numbers in brackets identifies a development task was performed by more than one individual.

Lesson	Sub-task	R - Hrs*	C - Hrs *	Hour Var.	U - Hrs *	Hour Var.
# 3	D [2]	17.5	20.0	2.5	15.7	- 1.8
(7 pages)	E [2]	11.0	10.5	- 0.5	8.7	- 2.3
G-9 A-6 V-1	M	11.0	11.3	0.3	14.5	3.5
	P	3.5	3.1	- 0.4	3.5	0.0
	T	1.5	1.7	0.2	1.7	0.2
Total		44.5	46.6	2.1	44.1	- 0.4
# 4	D [1]	24.0	22.2	- 1.8	18.0	- 6.0
(8 pages)	E [2]	24.2	13.2	-11.0	10.0	-14.2
G-15 A-7 V-0	M	12.0	13.0	1.0	16.7	4.7
	P	0.5	0.0	-0.5	4.0	3.5
	T	1.0	2.0	1.0	2.0	1.0
Total		61.7	50.4	-11.3	50.7	-11.0
# 9	D [2]	56.2	51.9	- 4.3	15.7	-40.5
(7 pages)	E [1]	6.7	7.7	1.0	8.7	2.0
G-2 A-5 V-4	M	17.5	18.4	1.0	19.2	1.7
	P	0.0	0.0	0.0	3.5	3.5
	T	1.5	1.7	0.2	1.7	0.2
Total		81.9	79.7	-2.2	48.8	-33.1
# 15	D [2]	42.0	39.4	- 2.6	15.7	-26.3
(7 pages)	E [1]	7.5	7.7	0.2	8.7	1.2
G-0 A-9 V-0	M	1.5	2.0	0.5	2.2	0.5
	P	0.0	0.0	0.0	3.5	3.5
	T	1.2	1.7	0.5	1.7	0.5
Total		52.2	50.8	-1.4	31.8	-20.4
# 17	D [1]	10.0	16.2	6.2	18.0	8.0
(8 pages)	E [1]	4.0	8.0	4.0	10.0	6.0
G-2 A-15 V-0	M	3.0	4.7	1.7	5.7	2.7
	P	0.0	0.0	0.0	4.0	4.0
	T	1.5	2.0	0.5	2.0	0.5
Total		18.5	30.9	12.4	39.2	20.7

Table 5: Comparison of reported, calculated and unadjusted staff-hours.

Hour variances are the expected hours minus reported hours. Negative values indicate an underestimation and positive values indicate an overestimation compared to reported hours.
*Reported Hours; Unadjusted Hours; Calculated Hours
Var. - Variance
Sub-tasks: D [Design]; E [Entry]; M [Media];
P [Programming]; T [Testing]
[1] [2] - Indicates a specific individual

Predicting effort for small projects (less than 2 staff-months) is a difficult endeavor. The estimation metrics presented in this paper are an attempt to predict effort for small courseware projects. The estimation baseline values need to be refined to overcome the short-comings that are identified in section 4.2. Due to the inaccuracy of the reported hours from the historical data, the baseline values were limited to data that could be validated as being reported correctly.

Lesson	Reported Hours	Calculated Hours	Hour Variance	Percent Variance
# 3	44.5	46.6	2.1	4.7
# 4	61.7	50.4	-11.3	- 18.3
# 9	81.9	79.7	- 2.2	- 2.6
# 15	52.2	50.8	- 1.4	- 2.6
# 17	18.5	30.9	12.4	67.0

Table 6: Percent Variance of Total Effort Hours

Comparing total calculated hours and unadjusted hours to reported hours [Table 6], calculated hours more accurately predicted actual reported hours than unadjusted hours for three of the five lessons, particularly in total calculated hours. For these lessons, the total hour variances (calculated hours - reported hours) ranged from an underestimation of 11.3 hours (lesson #4) to an overestimation of 12.4 hours (lesson #17). The greatest error variance can be seen in both the design (lesson #9) and entry task (lesson #4). In both cases, these tasks were performed by a less experienced staff member. The cost adjustment factors may need to be adjusted to compensate for lack of experience on two levels: the actual performance of the task and the processes used to perform that task. For instance, the entry task involves familiarity with the entry process and the template used to create the lesson.

4.2 Evaluation of the Estimation Model

The utility of a software cost model for practical estimation purposes can be evaluated using the following criteria [Boehm 1981]:

1. Definition. Has the model clearly defined the costs it is estimating,

and the cost it is excluding?

2. Fidelity. Are the estimates close to the actual costs expended on the projects?
3. Objectivity. Does the model avoid allocating most of the software cost variance to poorly calibrated subjective factors (such as complexity)? That is, is it hard to jigger the model to obtain any result you want?
4. Constructiveness. Can a user tell why the model gives the estimates it does? Does it help the user understand the software job to be done?
5. Detail. Does the model easily accommodate the estimation of a software system consisting of a number of subsystems and units? Does it give (accurate) phase and activity breakdowns?
6. Stability. Do small differences in inputs produce small differences in output cost estimates?
7. Scope. Does the model cover the class of software projects whose costs 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 well known until the project is complete?
10. Parsimony. Does the model avoid the use of highly redundant factors, or factors which make no appreciable contribution to the results?

The estimation method presented in this paper meets the criteria for definition, constructiveness, detail, stability, and scope, but does not completely fulfill the other criteria. Detailed evaluation of each criteria follows:

1. Definition. The scope has been restricted to estimation of all the sub-tasks of the development phase.
2. Fidelity. The total calculated hours are close to total actual hours reported for three of the five lessons created (within 4 staff-hours). For 4 of the 5 lessons, the estimate is within 20%, which is considered „reasonable“[Boehm 1981]. For individual sub-tasks, the estimates for media, programming and testing fulfills this criteria.
3. Objectivity. Unadjusted costs are allocated to the actual production of each task, but are subject to cost factors whose values are not well-defined.
4. Constructiveness. The estimation model employs the decomposition method which conveys the sub-tasks of the development phase. Understanding the sub-tasks needed to develop a lesson does not guarantee that the user will understand the details of performing each sub-task. The model was designed to be used by individuals who are familiar with the staff members assigned to the project. This criterion is met since the model explains the steps involved in calculating total effort.
5. Detail. This method relies on the activity breakdown of the development phase and can be used to estimate any of the sub-task activities individually.
6. Stability. Small differences in input values produce small differences in output cost estimations.
7. Scope. Since it was developed specifically for estimating

multimedia courseware, this model does cover the class of software projects whose costs need to be estimated.

8. Ease of Use. The unadjusted estimation of each sub-task is straight-forward, but the cost factor values are not easily determined since they rely on personal knowledge of the individuals involved in each sub-task.
9. Prospectiveness. Unless the user is familiar with all members involved in the development of a lesson, the cost factor values can not be accurately determined. This is particularly true in the design sub-task, with external subject matter experts cost factors and number of reviews to be conducted.
10. Parsimony. The estimation method does avoid the use of factors that are redundant or do not contribute appreciably to the result. Each sub-task has cost factors that will only effect estimation of that particular sub-task.

The effort distribution of courseware development differs from information systems [Figure 5]. The effort distribution for CBT in Figure 5 is based on the reported effort supplied for this article and reflects the varied tasks associated with CBT development.

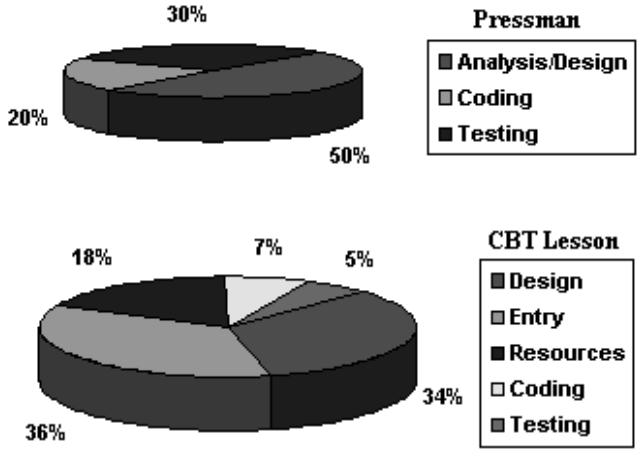

Figure 5: Development Effort Comparison: Information Systems and Courseware Development.

5 Conclusion

„The best criterion for the value of a metric is the degree to which it helps us make decisions“ [Boehm 1996]. Since every courseware house has different needs, development cycles, and development tools, it would be difficult to suggest one

specific metric or process to employ.

This paper presents an estimation method for calculating effort in the development of small multimedia CBT projects. The method is based on

(i) the decomposition of the development phase into sub-tasks and

(ii) using historical data as a baseline.

The use of this method is restricted to courseware projects using a pre-created template. It is not intended to be used by an individual unfamiliar with the staff members involved in the creation of the specific multimedia courseware or for projects that can not be defined by page units. Incorporation of calculated hours into a project management tool assists in the determination of realistic completion dates.

People tend to underestimate software size for three reasons [Boehm 1981]:

(1) People are basically optimistic and desire to please.

(2) People tend to have incomplete recall of previous experience.

(3) People are generally not familiar with the entire software job.

With baseline values derived from reported effort hours and sub-task estimates performed by the department or individual responsible for that sub-task, a more realistic total effort of development time can be calculated.

Refinement of the reporting system, particularly in the separation of the different types of media, may improve its baseline values. It is also interesting to explore replacing the experience cost factor with a baseline value of an individual staff member. This will allow the estimation model to adjust when new project teams are formed. Another enhancement would be to apply cost factors to each individual page, but the additional complexity may not justify the additional time required to calculate total effort.

Acknowledgments

Special thanks to Vivid Concepts for supplying the CBT development time reports that were essential for the creation of this project.

References

[AICC 1997] AICC: "Computer Managed Instruction Guidelines and Recommendations" from Aviation Industry CBT Committee Computer Managed Instruction, AGR 006, Version 1.1 by CMI Subcommittee at http://www.aicc.org/agr006.htm (1997)

[Boehm 1981] Boehm, Barry W.: "Software Engineering Economics", Englewood Cliffs, NJ, Prentice Hall (1981)

[Boehm 1991] Boehm, Barry: „Software Risk Management: Principles and Practice“, in IEEE Software, 8, 1 (1991), 32,41.

[Brooks 1995] Brooks, Frederick P., Jr.: "The Mythical Man-Month", Reading, MA, Addison-Wesley Publishing Company (1995)

[Fletcher et al. 1997] Fletcher, T., MacDonell, S.G., Wong, W.B.L.: „Early Experiences in Measuring Multimedia Systems Development Effort“ (1997), http://divcom.otago.ac.nz:800/COM/INFOSCI/SMRL/pub/pubs.htm.

[Grady 1994] Grady, R.: „Successfully Applying Software Metrics“, IEEE Computer, 27, 9 (1994), 18-25.

[Hackos 1996] Hackos, JoAnn:, „Estimating Risk in Instructional Design and Documentation projects“, Interactive Conference workshop (1996)

[Johnson 1990] Johnson, James R.: "The Software Factory: managing software development and maintenance", Wellesley, MA, QED Information Services, Inc. (1990)

[Jones 1996] Jones, Capers: „Our Worst Current Development Practices“, IEEE Software, 13, 2 (1996), 102-104.

[Marshall et al. 1995] Marshall, I.M., Samson, W.B., Dugard, P.I., Lund, G.R.: „The Mythical CourseWare Development to Delivery Time Ratio“, Computers Educ., 25, 3 (1995), 113-122.

[McConnell 1996] McConnell Steve: „Avoiding Classic Mistakes“, in IEEE Software, 13, 5 (1996), 112, 111.

[Pizzini et al. 1997] Pizzini, Q., Munro, A., Wogulis, J. Towne, D.: „The Cost Effective Authoring of Procedural Training“ (1997), http://btl.usc.edu/rides/shortPapers/costeff.html.

[Pressman 1992] Pressman, Roger S.: "Software Engineering, A Practioner’s Approach", New York, NY, McGraw-Hill, Inc. (1992)

[Rada 1995] Rada, Roy: "Developing Educational Hypermedia: Coordination and Reuse", Ablex Publishing: Norwood, New Jersey (1995)

[Rada 1997] Rada, Roy: Virtual Education Manifesto, Hypermedia Solutions Limited and http://hsl.gnacademy.org/gnacademy/hsl/ (1997)

[Rada and Schoening 1997] Rada, Roy and Schoening, James: "Educational Technology Standards" Communications of the ACM, 40, 9 (1997), 15-18.

[Symons 1990] Symons, Charles R.: "Software Sizing and Estimating MK II FPA (Function Point Analysis)", Chichester, New York, John Wiley & Sons. (1990)

[Stark et al. 1994] Stark, G., Durst, R.C., Vowell, C.W.: „Using Metrics in Management Decision Making“, IEEE Computer, 27, 9(1994), 42-48.

[Thackaberry 1997] Thackaberry, Christine W.: „Estimation Metrics for Courseware Maintenance Effort“, Master’s Degree Project, Washington State University. (1997)

[Veljkov 1990] Veljkov, Mark: „Managing Multimedia“, Byte, 15, 8 (1990), 227-232.

[Wasserman 1996] Wasserman, A. I.: „Toward a Discipline of Software Engineering“, IEEE Software, 13, 6 (1996), 23-31.