Examining the Feasibility of a Case-Based Reasoning Model for Software Effort Estimation

By: Tridas Mukhopadhyay
Graduate School of Industrial Administration
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213 U.S.A.

Steven S. Vicinanza
Energy Management Associates, Inc.
100 Northcreek
Atlanta, Georgia 30327 U.S.A.

Michael J. Prietula
Graduate School of Industrial Administration
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213 U.S.A.

Abstract

Existing algorithmic models fail to produce accurate software development effort estimates. To address this problem, a case-based reasoning model, called Estor, was developed based on the verbal protocols of a human expert solving a set of estimation problems. Estor was then presented with 15 software effort estimation tasks. The estimates of Estor were compared to those of the expert as well as those of the function point and COCOMO estimations of the projects. The estimates generated by the human expert and Estor were more accurate and consistent than those of the function point and COCOMO methods. In fact, Estor was nearly as accurate and consistent as the expert. These results suggest that a case-based reasoning approach for software effort estimation holds promise and merits additional research.

Keywords: Software effort estimation, case-based reasoning, constructive cost model, function points

ACM Categories: D.2.1, D.2.2, D.2.8, D.2.9, D.3.2, H.1.0, K.6.3

Introduction

Software is expensive to develop and is a major cost factor in corporate information systems budgets. The magnitude of software investments, estimated at more than $200 billion annually (Boehm, 1987), forces management to carefully consider costs and benefits before committing the required resources to any potential software development project. Naturally, the accuracy of software project estimates has a direct and significant impact on the quality of the firm’s software investment decisions.

When costs are underestimated, some projects are undertaken with an inflated impression of their worth to the firm given the actual costs (i.e., effort) to develop them. Projects originally thought to be valuable may not subsequently prove to be cost-effective. Up to 15 percent of new development projects are abandoned mid-stream, largely due to cost overruns (Jones, 1986a). Underestimated projects that do reach completion are often released prematurely to meet the budget; these may omit important features or system testing and result in incomplete and unreliable systems (Kemerer, 1989).

When costs are overestimated, inflated project estimates may actually increase the project cost by putting less pressure on programmers to be productive (Abdel-Hamid and Madnick, 1986). Additionally, projects possessing a real potential for benefit may be mistakenly rejected as too expensive, resulting in the cost of a missed opportunity to create value within the firm.

Consequently, both overestimation and underestimation may result in costly errors. Accurate project estimation can reduce these unnecessary costs and thereby increase the firm’s efficiency (e.g., by making more appropriate resource allocation decisions of programmers’ time) as well as its effectiveness (e.g., by selecting more
appropriate projects through improved cost/benefit analysis. As software becomes increasingly expensive and critical to today's organizations, the consequences of estimation errors become equally significant, further underscoring the need for accurate estimation techniques.

Methods of improving estimations have, for the most part, been based on analytical models. A number of such approaches have been generated by researchers (Cote, et al., 1988). However, attempts at validating them have been largely unsuccessful. For example, uncalibrated models may average as much as 600 percent relative error (Kemerer, 1987). Even with local calibration and adjustment, their use in industry is often restricted to the verification of estimates generated by manual techniques (Zelkowitz, et al., 1984).

We propose that the analytical approach is not wrong but is insufficient. We suggest that qualitative improvements in estimation accuracy will not come from the application of analytical models alone. Additional insights must be obtained from other sources. One such source is the people who successfully adapt to the demands of the estimation task—the experts. Previous research has suggested that expertise at estimating software effort does exist and accurate estimates can be generated by highly experienced software development managers (Vicinanza, et al., 1991). The most accurate expert in that study relied upon a distinct form of reasoning to solve estimation problems. In particular, the expert utilized a form of analogical problem solving called case-based reasoning in which effort estimation was driven by recall of previously encountered software projects. The purpose of this article is to construct a computational model based on the case-based reasoning strategy used by the most accurate human estimator and examine the extent to which the model can generate accurate estimates.

The case-based approach used is often encountered in task domains that have no strong theoretical model and where the domain rules are incomplete, ill-defined, and inconsistent (Ashley and Rissland, 1987). Viewed from the perspective of task adaptation, the structure of a task may not support the generation of knowledge reflecting deep, causal principles for its resolution; rather, the structure may require more surface-level, exemplar-based knowledge. Problems are solved in such domains not by finding and applying the knowledge of the most appropriate fundamental principle but by finding and applying the knowledge of the most relevant prior case (Prietula, et al., 1989). Because the domain of software effort estimation lacks a strong causal model based on deep principles and is situated within an often-changing, highly context-dependent task environment, we propose that the case-based approach evidenced by the expert is indeed an appropriate suite of adaptive mechanisms to bring to bear on the problem.

We developed a case-based reasoning model, called Estor, incorporating five basic analogical problem-solving processes: problem representation, analog retrieval, solution transfer, attribute mapping, and non-correspondence adjustment. These five processes were supplemented with the domain-specific knowledge of the referent expert. We then subjected the model to a test in which the accuracy of Estor's estimates are compared to those of function points (Albrecht and Gaffney, 1983), COCOMO (Boehm, 1981), and the human expert on a common set of estimation problems. Our results showed that Estor approaches the accuracy and consistency of the expert and exceeds the accuracy and consistency of other methods. However, given the limited sample size used, these results should realistically be viewed as an indicator of the plausibility of this approach.

We first summarize the current key research on software effort estimation. Next, we discuss the theory underlying the case-based reasoning approach. We then describe Estor and report on a test in which the accuracy of Estor's estimates is compared to those of other methods. We conclude with a discussion of the implications of our research.

**Estimating Software Costs**

A fundamental problem of software effort estimation is the determination of software size. As explained later, software estimation is also complicated by several productivity factors.
searchers have taken two main approaches to measure software size: lines of code (LOC) and function points.

**LOC-based models**

A widely known model based on LOC is the constructive cost model (COCOMO) (Boehm, 1981). COCOMO classifies projects into three broad categories: organic or simple, semidetached or average, and embedded or complex. The COCOMO model itself has three versions. The intermediate version of COCOMO first calculates a nominal effort estimate in worker-months (WM) using a non-linear function based on the size of the software measured in thousands of delivered source instructions (KDSI):

$$WM = \alpha (KDSI)^{13}$$

where the values of the constants $\alpha$ and $\beta$ are different for organic, semidetached, and embedded projects. Next, it adjusts the nominal estimate by multiplying WM by the ratings on 15 "cost drivers" that include attributes of the product, computer, personnel, and project. The COCOMO basic model, however, does not use any cost drivers. The COCOMO detailed model, on the other hand, divides the project into four phases (product design, detailed design, coding/unit test, and integration test) and estimates and applies the 15 cost drivers to each phase separately rather than to the entire project. Other models based on non-linear functions of LOC include the Doty model (Herd, et al., 1977) and the meta-model by Bailey and Basili (1981).

Another set of LOC-based models uses standard distributions as the basis for modeling the phase distribution of effort. Putnam's SLIM model, for example, determines the life cycle effort ($K$) in worker years based on number of source statements ($S$) (Putnam, 1978):

$$K = S^3 C^{-3} t_d^{-4}$$

where $t_d$ represents the time of peak manpower deployment and $C$ is a technology constant. SLIM uses the Rayleigh curve to model the distribution of effort over time. The Jensen model also uses the Rayleigh curve for effort distribution (Jensen, 1983).

A criticism of the LOC-based models is that they require estimating LOC before development begins. However, accurate LOC estimates may not be available until after the detail design is complete. The focus on LOC as an indicator of size also leads to problems when a model calibrated for one coding language is used for another without recalibration. Finally, variations in line counting methods may change LOC by a wide margin (Jones, 1986b).

**Function point-based models**

The function point method first assigns a weight to each unique input type, output type, logical file, external interface file, and external query handled by an application to reflect the "level of complexity." The total score for all function types, called the function count, is then modified using the total ratings (TR) of 14 processing complexity characteristics to account for the different kinds of system requirements and development environments:

$$\text{Function Points} = (\text{TR} \times .01 + 0.65) \times \text{Function Counts}$$

where TR ranges between 0 and 70. The major advantages of the function point method are the feasibility of estimating system size at an early stage and the independence of function points from the coding languages.

A simple linear regression can be used to estimate person-months as a function of function points (Albrecht and Gaffney, 1983). The function point approach has also been adapted to generate new models. For example, the ESTIMACS model uses a modified function point method for a size estimate that is subsequently adjusted by assumptions about project complexity (Rubin, 1983). Another model derived from the function point approach is the SPR/100 model (Jones, 1986b). This model includes 175 product and process-related variables, although any specific estimate typically uses between 50 and 100 variables.

Although the function point approach can provide an early estimate of size, it also has certain problems. The two dominant problems associated with this metric involve the effort required to collect function point data and the difficulty in obtaining consistent estimates from multiple individuals (Kemerer, 1989). In addition, function point measure requires rescaling for real-time.
systems and database environments (Symons, 1988).

**Empirical evaluation**

Given the wide variety of models available today, a question naturally arises: How good are these models? Using a large data set collected over 10 years, one study found little support for the strict time-effort relationship suggested by the models based on the Rayleigh curve (Jeffery, 1987). A second study employed a data set of 20 projects to evaluate both SLIM and COCOMO models and found little systematic relation in the plots of actual versus estimate (Kitchenham and Taylor, 1985). Yet another study found that COCOMO and SLIM tend to considerably overestimate data processing applications (Kemerer, 1987). In comparison, the ESTIMACS and function point models performed much better.

There are important reasons why these models may not perform well. First, existing models lack a solid theoretical base to go beyond statistical associations. No causal models exist within these approaches to explain a generated estimation. Second, estimating software size is an inherently difficult problem. Finally, the calibration of these models is difficult in a real-world setting. Given the difficulties with existing models, it is not surprising that their practical use is limited as a supplement to other methods (Zelkowitz, et al., 1984).

Accordingly, most alternative methods require the use of human expertise in one form or another to estimate development cost (Boehm, 1984). Although the use of expert judgment is commonplace in industry (Fairley, 1985; Wrigley and Dexter, 1987), few researchers have examined this approach. In the most direct examination of this method to date, a recent study reported that some experts, given input parameters to COCOMO and function point models, estimated effort with very low error compared to either the COCOMO or the function point model (Vicinanza, et al., 1991). The experts also proved to be much more sensitive to factors affecting productivity than either COCOMO or function points, as evidenced by markedly higher correlations between the experts' estimates and actual project efforts. It was concluded that the most accurate estimator relied on a particular kind of analogical strategy—case-based reasoning. Thus, we focus on modeling this problem-solving method.

**Mechanisms of Analogy and Case-Based Reasoning**

Analogical reasoning is a fundamental tool in the human problem-solving repertoire (Sternberg, 1977). Although widely recognized as a useful method for software cost estimation, little theoretical or empirical work has been reported supporting the use of analogy in this domain. Analogy is the primary method NASA systems designers use to estimate the size and execution time of new ground-based satellite control systems (Silverman, 1985). The analogical approach has also been applied to the development of knowledge bases describing orbital trajectory simulation systems (Allen, 1990). The empirical literature on analogical reasoning in software estimation is, however, virtually non-existent.

At the most general level, analogical problem solving involves relating some previously solved problem or experience to a current, unsolved problem in a way that facilitates the search for an acceptable solution. The problem to be solved is referred to as the target of the analogy. The previous problem is called the source of analogy. The formation of the analogy occurs when there is a perceived similarity between the source and target whose basis is dependent upon the problem-solving domain context. Similar elements between the source and target are mapped to one another. This mapping is then used to form analogical inferences that generate information to facilitate problem solution.

General theories of analogical problem solving describe frameworks for understanding the processes that an expert using this type of reasoning should exhibit while developing project estimates. Nevertheless, these theories, broad in scope and covering a wide range of analogical reasoning situations, are too general for our purposes. However, a more specific framework for studying analogical problem solving proposed by researchers building computational models of case-based reasoning is relevant to our work.\(^5\)

---

\(^{4}\) See, for example, Benbasat and Vessey (1980), Boehm (1984), Fairley (1985).

Whereas a general theory of analogical problem solving must accommodate the mapping of a source analog whose elements are conceptually remote from those of the target (i.e., across task domains), research into case-based reasoning has regularly focused on situations where the source analog is drawn from the same general problem domain and has the same representational structure as the target. By constraining the source of the analogy to previously encountered cases of the same problem class, the case-based framework provides a more explicit definition of the underlying cognitive processes we should expect to find in software cost estimation. In this domain, the representation of the target case, a software project, is similar to that of previous cases—earlier software projects. Thus, the generic mechanisms of analogy may be viewed as a general but “weak” method of problem solving (Newell, 1969). However, the within-domain refinements (via task-specific representations and knowledge) afford particular adaptations that seem to yield superior performance in software estimation problems.

In case-based reasoning, the problem solver, after creating a mental representation of the target problem, retrieves from long-term memory one or more previous problem-solving episodes, or cases, that have similar features. These cases are then evaluated, and the most appropriate one is selected as the source analog. The mapping of source-to-target features is often fairly straightforward because a common set of features is shared among all cases. The solution that achieved the goal in the source problem is then transferred to the target and subsequently modified to compensate for analogical elements whose mappings are not in correspondence.

With guidance from analogical and case-based theories of problem solving, a framework for a detailed study of expert reasoning at software estimation is defined in the next section.

**Estor: A Case-Based Reasoning Model**

In a prior study, problem-solving data were collected from highly experienced software managers (Vicinanza, et al., 1991). Each manager estimated the effort required to complete each of 10 software development projects. The software projects used in the study came from Kemerer (1987) and comprised 37 project factors as well as the actual development effort associated with each of the 10 completed projects.

Based on the accuracy and consistency of the estimates, one subject was selected as the “referent expert” for the construction of Estor. This individual was a software development manager at a large software company with 10 years of total software management experience and nine of those years in data processing applications similar to the problem set projects.

The tape-recorded protocol of the referent expert solving 10 estimation problems was subjected to a process tracing analysis for the knowledge acquisition task (Ericsson and Simon, 1984). Using the case-based reasoning form of analogical problem solving as a theoretical model, protocol segments were classified into the following five process categories:

1. Construct—develop a representation of the target problem.
2. Retrieve—select and retrieve a source analog.
3. Transfer—use the solution of the source analog as the starting point.
4. Map—find the differences between the source and target problem.
5. Adjust—Adjust the initial solution based on the differences found.

The utility of using this theoretical structure as a guide is that it enables the discrimination of a general cognitive mechanism (though one adapted for analogical functioning) from the knowledge needed to apply that mechanism to a specific domain. The computational model, Estor, implements the five processes in a domain-independent form, while the domain-specific knowledge obtained from the protocol analysis is contained in a separate knowledge-base. Three distinct types of domain knowledge comprise that knowledge-base: representations of cases (i.e., software projects), knowledge to select an appropriate analog for each target case, and knowledge to adjust the estimate based on the interaction between the source and target case representations. We first discuss the generic case-based processes implemented by Estor and then describe the structure of the knowledge base.
Implementation of the case-based reasoning strategy in Estor

Estor employs the five basic processes: construct, retrieve, transfer, map, and adjust. The logical architecture for Estor is presented in Figure 1.

The *construct* process is used to create an internal representation of the target problem. For each project, the project's attributes are manually typed into the system and stored in individual project schemas.

To *retrieve* an appropriate source analog, Estor invokes a domain-specific case selection heuristic. The heuristic examines each project in the case knowledge-base and selects one as the source analog based on a “similarity distance” (described later) to the target.

The next step is the *transfer* of the solution that achieved the goal in the source case to the target case. In Estor, solution transfer is accomplished by referencing the effort attribute of the source project and transferring it to the effort attribute of the target project schema.

Estor *maps* the source and target by bringing each attribute of the source and target one by one into working memory, comparing them, and adding non-corresponding attributes to a list kept in memory.

To *adjust* the estimate for a non-corresponding attribute, Estor uses production rules via an interpreter written specifically for Estor. The interpreter creates a conflict set of rules in the rule base that will adjust for each non-corresponding attribute in the list. If the set contains more than one rule, then the conflict is resolved using the specificity principle, which gives preference to the rule with a precondition set more specific to the current situation (Anderson, 1983).

Software estimation domain knowledge

There are three distinct classes of domain knowledge required by Estor to estimate effort using its inference mechanism: case knowledge, case selection knowledge, and non-correspondence adjustment knowledge.

*Case knowledge* reflects the episodic memory of previously encountered software projects, including the actual amount of effort required for each one. The project abstraction created for this study comprises function point and COCOMO inputs. Because of the highly contextual and bundled nature of this knowledge, the computational representation scheme selected was the schema or frame (Minsky, 1975).

The values of project attributes for source cases were inferred from the verbal protocols. The expert relied primarily on three different source cases, two of which were successfully reconstructed from the protocols. These two projects constituted the “paradigm” cases for subsequent estimation tasks (Hunter, 1989a).

*Case selection knowledge* permits the selection of an appropriate analog from the set of paradigm cases. The protocol analysis indicated that the expert used a combination of available size metrics to make this selection, but the verbalization of this process in the protocol was inadequate for its reconstruction. Because case retrieval and selection are intimately linked to the organization of episodic memory and employ subconscious cognitive processes (Shank, 1982), it would be difficult, if not impossible, for the subject to directly provide an accurate description of the cognitive processing invoked by this task. Thus, Estor incorporates a plausible combination of the available function count data to discriminate among candidate source analogs. The function count “similarity distance” is measured by the sum of squares difference in five function count components between the target and candidate source projects. The project that minimizes this difference is selected as the source analog.

Finally, *adjustment knowledge* is brought to bear when non-correspondences arise in mappings between attributes of the source and target projects. In contrast to case knowledge, this knowledge has a smaller granularity, is less context-specific, and requires particular invocation considerations, so interpreted production rules provide an acceptable computational mechanism. The preconditions of these rules are sensitive to those attributes for which the source and analog projects have different values. After

---

4 This is generically called the “indexing problem,” and although interesting general research is proceeding (e.g., Hunter, 1989b; Kolodner, 1989; Martin, 1989), the results are not yet unequivocal or generalizable to any given domain.
the non-corresponding project attributes are discovered, rule actions make the appropriate analogical inferences by adjusting the target project estimate to compensate for differences between the projects.

Rule "preconditions" were inferred from protocol statements. Preconditions were generalized so they would apply in similar, though not necessarily identical, situations. For example, assume a mapping in which source project reliability is "high" and target reliability is "average." Rather than using these specific values, the resulting rule is crafted to fire if the reliability difference between two projects is one semantic unit on the relevant underlying scale for the attribute (e.g., low, medium, high, very high). Similarly, rule "actions" were coded from statements in the protocol. The magnitude of the adjustment was generalized as a percentage of the source project estimate rather than an absolute number of worker-months as specified in the protocol. For example, the following production rule (expressed in English) increases effort estimate (by 100 percent of the source project effort) due to the non-correspondence in project complexity levels:

```
IF Complexity of the target project is two semantic units more than that of the analog, THEN Increase the effort estimation by adding back an amount equal to the effort of the analog.
```

An estimation example
To illustrate the approaches, consider project 1 from Kemerer (1987) that took 277 worker-months to complete. We first summarize the approaches taken by the referent expert and by Estor. We then solve this problem using both COCOMO and function point methods.

1. Expert Estimate: The expert examined the 37 project attributes one by one and then concentrated on the five function count attributes. Next, the expert recalled a purchase order system previously worked on and decided to use the effort required for that system, 90 worker-months, as the starting point. The expert then examined the attributes of the project and, for seven of them, found the problem had a different value from the analog (i.e., the purchase order system) and adjusted for
these differences. The expert’s final estimate was 250 worker-months.

2. Estor Solution: Estor first constructed an internal representation of the target case using the project attributes and then retrieved source case 1 as the analog based on the function count similarity metric. Estor transferred the effort of the analog (90 worker-months) to the target case as its starting solution. Next, Estor mapped the attributes of the source and target cases and found seven non-corresponding attributes. Eight production rules could possibly fire, but two of these were abandoned because they were less specific compared to the other rules. Each rule adjusted the initial estimate upward or downward, resulting in the final estimate of 287 worker-months. Note that in this example, Estor used the same source case as the expert and found the same number of non-corresponding attributes but did not use the same set of rules as the expert. A complete analysis of the divergence between the expert and Estor will be discussed later in this article.

3. COCOMO Estimate: We illustrate the COCOMO model using its intermediate version. The development mode of this project was considered semidetached. Accordingly, a nominal estimate of 1,478.3 worker-months was generated based on the project size 253.6 thousand delivered source instructions. Next, the project was analyzed using 15 attributes. Whenever the project deviated from the nominal value of an attribute, the nominal estimate was adjusted using the appropriate factor. The final project estimate was 1,238.6 worker-months.

4. Function Point Estimate: First, each unique input, output, logical file, external interface file, and external query was identified. Each of these function types was given a score to reflect its level of complexity. The total score 1,010 was the function count of the project. Next, the total ratings (TR) of 14 processing complexity characteristics was summed to 55.5. The function point was then calculated as 1,217.1 by modifying the function count by TR. Finally, using the linear formula developed by Albrecht and Gaffney (1983), the effort estimate was found to be 344.3 worker-months.

Comparison of Performances

A comparison of the performances of the referent expert and Estor was done in absolute and relative terms with respect to each other and to two well-known approaches (COCOMO and function points). The points of divergence between the expert and Estor in solving the estimation problems were also examined. Finally, the impacts of the three components of the knowledge base on Estor’s performance were explored in this study.

Accuracy and consistency

Based on the software effort estimation literature, two criteria were used to compare Estor with other methods of estimation: accuracy and consistency. The accuracy of a method is important because the success of a software development project is directly affected by the effort estimation. Inaccurate estimates can lead to the selection of wrong projects and may impact project scheduling and staffing decisions.

The other important property of software effort models is consistency. An inaccurate model can be consistent if it uniformly misestimates effort for a set of projects. For example, a model that consistently overestimates effort by 50 percent is more desirable than a second model that randomly overestimates effort by 50 percent for half the projects and underestimates by 50 percent for the rest. If a model is not consistent, then management may not place much trust on its estimates. As a result, an inconsistent model may be of little use in practice.

It is expected that the four methods (COCOMO, function point, the expert, and Estor) are not equally accurate or consistent due to their intrinsic differences. First, the expert is expected to surpass the other methods (including Estor, because Estor is not a complete representation of the expert’s knowledge base). Second, Estor is expected to outperform COCOMO and function point methods if it models sufficient aspects of the expert’s reasoning process. Finally, based on prior work, function point is expected to be more accurate, if not more consistent, than COCOMO (e.g., Kemerer, 1987).

---

It is difficult to specify a rank order for the consistency property of the four methods. In particular, both theoretical and empirical evidence are scant on which of the two algorithmic models, COCOMO and function point, is more consistent. However, it is expected that the expert and Estor are more consistent than either COCOMO or the function point model. This expectation is based on the domain-specific adjustment knowledge available to the expert, and possibly to a lesser degree to Estor, that enables effective compensations for the impacts of productivity factors in the development process. (See the Appendix for formal statistical hypotheses tested.)

Materials and procedure

The estimation projects in our study included 15 projects representing medium to large data processing systems, ranging in size from 39,000 to 450,000 LOC (100 to 2,300 function points) and requiring from 23 to 1,107 worker-months of effort to complete (Kemerer, 1987). Most of the projects were written in COBOL and developed for IBM mainframe systems.

Estor provided estimates for the 15 estimation problems. In addition, estimates of these problems were also obtained from a COCOMO and function point analysis (Kemerer, 1987). The referent expert had solved 10 of these problems in a prior study (Vicinanza, et al., 1991). The verbal protocols from these 10 projects constituted the source of domain knowledge of Estor. The expert solved the five remaining estimation problems for this study.

Measures

Accuracy

One obvious measure of accuracy is the magnitude of the difference between the estimate and the actual development effort. However, using the absolute difference between the two values is problematic. An error of 20 worker-months on a 1,000 worker-months project is considered a trivial error. On the other hand, the same error for a 20 worker-months project is quite serious.

A more useful error metric that compensates the absolute error for project size is percentage error. A problem that persists with percentage error is that over-estimates and under-estimates tend to cancel each other out when estimates of multiple projects are averaged together. One final adjustment is required, which is to use the absolute value of the percentage error. This number has been termed magnitude of relative error or MRE. The MRE is calculated for a project by the following formula:

\[ MRE = 100 \frac{|Actual Effort - Estimated Effort|}{Actual Effort} \]

Note that accuracy is inversely proportional to the MRE of a project. The accuracy of an estimation method can be evaluated using the average MRE for a set of projects.

Consistency

A model that is sensitive to the influence of various productivity factors may nonetheless consistently overestimate or underestimate development if the standard productivity rate assumed by the model is significantly different from that of the environment in which the software is developed. One measure of the sensitivity of the model to development factors that is not dependent on the base productivity rate is the correlation between the estimates and the actuals. When there is a strong correlation, larger projects are estimated as requiring more effort to complete than smaller projects. Thus, the consistency of a method is measured as the correlation coefficient between actual effort and estimated effort across a sample of projects.

Data analysis

The data set consisted of actual and estimated effort figures for COCOMO, function point, the expert, and Estor for 15 projects. This data set was divided into two subsets. The first subset contained the data for the first 10 projects, while the second comprised the remaining projects. Dividing the data set into these two subsets takes into account that Estor was developed based on the verbal protocols of the expert solving the first 10 projects.

---

6 MRE was suggested by Thebaut (1983) and Conte, et al. (1986) and was used by Kemerer (1987) in his validation study.
7 Several researchers have used the correlation measure to check the level of consistency (Albrecht and Gaffney, 1983; Behrens, 1983; Kemerer, 1987).
It was expected that Estor's estimation results from the two subsets would not be significantly different. Estor would not be architecturally different if it were based on the verbal protocols of all 15 projects because it would still use the same two source cases as analogs. Estor would still use the same selection heuristic. It would, however, have additional rules derived from the last five projects. The resulting impact is not certain because rules in Estor were not taken verbatim but generalized from the verbal protocols.

Before undertaking the data analyses, Estor's performance figures for the 10-project subset and the full data set were compared (see Table 1). Both the average MRE and the correlation between actual effort and estimated effort of Estor were very similar for the two data sets. Thus, the full data set was used for our analysis.

A statistical test was used to compare the accuracy of the four methods (see the Appendix for more detail). The test ranked the methods according to the accuracy of the estimates generated and compared the total rank for each method for all projects. To study the consistency of the methods, pair-wise statistical comparisons of correlations were made. For the four methods, six such comparisons were made.

Two more analyses were performed as part of this study. A qualitative analysis was performed to compare the problem-solving behavior of Estor against that of the expert. Finally, an exploratory analysis was undertaken to understand the impact of each of the three components of the knowledge base of Estor on its performance. The goal of this analysis was to gauge how the accuracy and consistency of Estor change as each component of the knowledge base is modified.

**Results**

This section reports the results of three analyses: accuracy and consistency of estimates, points of divergence between the expert and Estor, and a sensitivity analysis of Estor.

**Accuracy and consistency of estimates**

For all estimators, two primary data were used in the analysis: effort estimates and derived MREs. Table 2 summarizes the MRE results for each estimator, including the extreme MRE values, the standard deviation, and the mean. As the table illustrates, by MRE measures the performance of Estor was not quite as good as the human expert but better than either of the algorithmic models.

We predicted that the accuracy of the expert would be highest, followed by Estor, function point, and COCOMO. Consistent with our expectations, the statistical tests supported our ranking of the four methods.

Table 3 summarizes the results of the correlation analysis for this data set. Our statistical results indicate no difference between the expert and Estor and no difference between COCOMO and function points. However, both the expert and Estor had significantly higher correlation than either COCOMO or function points. Thus, the expert and Estor are equally consistent, and both are more consistent than either the function point or COCOMO method.

**Points of divergence between the expert and Estor**

A qualitative analysis identified two points of divergence between Estor and the referent expert by comparing aspects of the protocols generated during problem solving. The first difference between the two was in the retrieval of the source analog—the underlying case schema. The expert was able to use more of the information about both the target and candidate sources when retrieving a prior case from memory. As previously noted, the verbal protocols contained little explicit information about the cognitive processes involved in selecting a source project. Estor's retrieval heuristic, based primarily on function counts, was chosen by determining which project factors the subject was most often considering immediately before retrieving the source project. How the expert used this information and any additional information that was examined could not be determined from the verbal protocols. Consequently, the selection heuristic in Estor did not always make the same choice of source analog as did the expert, resulting in different estimates because of subsequent differences in the initial base effort and non-correspondence mappings.

A second point of divergence between the two was in the selection of adjustment rules. Estor
Modeling Effort Estimation

Table 1. Estor's Performance Figures

<table>
<thead>
<tr>
<th></th>
<th>First 10 Projects</th>
<th>All 15 Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average MRE</td>
<td>50.62%</td>
<td>52.79%</td>
</tr>
<tr>
<td>Correlation between actual effort and estimated effort</td>
<td>0.975</td>
<td>0.970</td>
</tr>
</tbody>
</table>

Table 2. MRE Results for Each Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Average MRE</th>
<th>Std. Dev. (MRE)</th>
<th>Maximum MRE</th>
<th>Minimum MRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert</td>
<td>30.72</td>
<td>21.74</td>
<td>72.41</td>
<td>0.86</td>
</tr>
<tr>
<td>Estor</td>
<td>52.79</td>
<td>37.92</td>
<td>106.90</td>
<td>0.98</td>
</tr>
<tr>
<td>Function Point</td>
<td>102.74</td>
<td>116.05</td>
<td>326.72</td>
<td>0.23</td>
</tr>
<tr>
<td>COCOMO</td>
<td>618.99</td>
<td>680.83</td>
<td>2,685.34</td>
<td>106.97</td>
</tr>
</tbody>
</table>

Table 3. Correlation Between Actual Effort and Estimated Effort

<table>
<thead>
<tr>
<th>Estimator</th>
<th>Correlation</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert</td>
<td>.985</td>
<td>.001</td>
</tr>
<tr>
<td>Estor</td>
<td>.970</td>
<td>.001</td>
</tr>
<tr>
<td>COCOMO</td>
<td>.825</td>
<td>.01</td>
</tr>
<tr>
<td>Function Point</td>
<td>.742</td>
<td>.01</td>
</tr>
</tbody>
</table>

sometimes applied rules to mapping non-correspondences that were missed by the expert, resulting in more accurate estimates than those of the expert. Unfortunately, this situation was more than compensated for by cases where there were insufficient rules in the knowledge base to make the needed analogical inferences. In these cases, Estor's estimates were much less accurate than those of the referent expert.

The identification of these points of divergence was important because it showed where the system could be modified to improve its performance. From the initial analysis, three basic improvements would be (1) a larger rule base to handle adjustments; (2) a larger (though representative) case base; and (3) a better selection heuristic. Next, we examine the sensitivity of Estor to each of these three components of its knowledge base.

Sensitivity analysis of Estor

As stated earlier, the purpose of the sensitivity analysis was to determine the relative contributions of the rule base, case base, and case selection heuristic to Estor's performance. Recall that the average MRE for Estor was 52.8% (Table 2), and the correlation between actual effort and Estor's estimates was .97 (Table 3).

Rule Base Sensitivity

To gauge Estor's sensitivity to the adjustment rules, the rule base was disabled and estimates were generated for the entire problem set. That is, Estor made effort estimation for each project based solely on the paradigm case it retrieved and made no subsequent adjustments. The resulting average MRE increased slightly to 65.1%. However, the correlation between the ac-
tual and estimated effort revealed a more drastic reduction, dropping to .75. These results indicate that the rule base in Estor enhances its consistency more than its accuracy.

Case Base Sensitivity
For this analysis, Estor had the entire rule base available for adjustments but was forced to generate estimates from a particular paradigm case. Therefore, two sets of estimates were generated: one set for each of the two paradigm cases available. When using Paradigm Case 1, the average MRE increased to 78.2%, and the correlation declined to .95. When using Paradigm Case 2, the average MRE increased to 72.5%, and the correlation declined to .91. Interestingly, the effect of forcing a particular paradigm case had almost the complementary effect of disabling the rule base on Estor's performance. These results indicate that the case base in Estor enhances its accuracy more than its consistency.

Case Selection Heuristic
To improve our understanding of the role of the selection heuristic in Estor, we determined, for each of the 15 projects in the problem set, the MRE scores using each of the two paradigm cases as the source case. For each project, the “best” heuristic would select the paradigm case that would lead to the lower of the two MRE scores. Our results showed that such a heuristic would lead to an average MRE of 40.6% (a decrease of 12.2% in Estor’s average MRE), whereas the “worst” heuristic would result in an average MRE of 111% (an increase of 58.2% in Estor’s average MRE). Thus, the selection heuristic used by Estor appears to be reasonable for the cases in the problem set. Furthermore, the selection mechanism can have a considerable effect on the performance of Estor. In summary, a higher level of performance can be achieved by Estor if improvements are made in any of the three components of the knowledge base.

Discussion
In general, the attributes of a task environment will favor some problem-solving strategies over others (Simon, 1981). In this instance, it appears that case-based analogical problem solving is well-suited to the demands of the cost estimation problem environment. As noted earlier, case-based reasoning seems to emerge as a successful strategy in domains where the rules are ill-defined and there is no strong theoretical model to guide problem solving. A number of reasons explains the lack of a strong theoretical estimation model for software estimation. One is the large number of project factors. A prior study identified 74 different factors used by effort models (Wrigley and Dexter, 1987). Many of these factors are subjectively defined, and even those that appear to be well-defined, such as LOC, are not what they seem (c.f. Jones, 1986b). Furthermore, many factors have nonlinear effects and covary, further complicating the construction of quantitative models.

Given these obstacles, why might a case-based approach succeed in this domain? The answer may be found in the nature of the case-based approach. A case-based reasoner will perform optimally when the source and target projects are identical. In this situation, there will be no non-correspondences and the estimate will be identical to the source effort. However, as the difference between the source and target projects increases, non-correspondences will arise that require adjustments to the estimate. Because the adjustment rules themselves are heuristics, each adjustment may add some amount of error or uncertainty to the estimate. Case-based reasoning was effective in this study because the paradigm cases available to Estor were appropriate for the problems presented (i.e., data processing applications). Thus, error was significantly reduced by minimizing the number of non-correspondence adjustments.

While Estor has been shown to be useful, it is constrained in that it has been developed from a subset of potential software development factors—in particular, those of the function point and COCOMO models. It is likely that other important factors can, and should, be incorporated into this model. However, this constraint is not so much a deficit of the model as a gap in the knowledge base, which may be remedied through subsequent research.

There are several contributions made by this study. First, a theoretical model of analogical reasoning has been applied to the domain of software cost estimation. This research demonstrates the processes of analogical reasoning in
a problem domain and specifies which domain-specific knowledge structures are sufficient to apply this problem-solving strategy to the estimation of project development effort. The domain-specific knowledge has been identified and distinguished from the mechanisms underlying the basic reasoning process.

The strengths of Estor complement those of the human estimator. It is clear that human memory is fallible—past projects can be forgotten, attributes confused, non-correspondences missed, or important factors inadvertently skipped. Estor's memory, however, is immune to decay or distortion. It can perform an exhaustive search of the case base for the best analog. On the other hand, Estor cannot generate adjustment heuristics, unlike the human expert. The combination of the general domain knowledge of the expert with the precision and speed of Estor's memory and analogical mapping mechanism may lead to a powerful synergy. As it is currently configured, Estor might best be utilized not as an expert system but rather as an expert support system—a workbench that a development manager can use as an aid in estimating software effort in the development process. It is conceivable that Estor will be loaded with site-specific paradigm cases to capture the base productivity of the development group. The development manager should also be able to formulate new rules and test their efficacy against the existing rules in Estor. Practice of this nature may evolve into a simple mechanism that enriches Estor from the continual learning at the development site.

For Estor to become an expert support system, some of its current limitations have to be overcome. It is clear that the domain knowledge of Estor requires improvement. In fact, Estor cannot claim to be perfect psychological model of the expert's problem-solving method and knowledge.

A number of issues need to be considered in this respect. What attributes would the expert use if he or she were not constrained by the inputs of the COCOMO and function point models? What rules would the expert use if he or she had all attributes of his or her choice? Which attributes are most important for Estor's adjustment knowledge? What would be the impact on Estor's performance if the number of paradigm cases were increased? What should be the appropriate number of paradigm cases? Which selection heuristic would retrieve the suitable paradigm case?

Finally, because the sample size for the analyses presented here is limited, the current validation of Estor's performance can only be viewed as an indicator of plausibility rather than an unequivocal verification of generalizability. Further studies will be necessary to validate the more general applicability of this approach.

Conclusion

This study has presented a model of case-based analogical software cost estimation and has described an example of that model in the form of a computer program called Estor. We have demonstrated the plausibility of case-based reasoning as a form of problem solving in this domain and have illustrated the potential for improving the accuracy of software cost estimates through this form of deliberation. Estor did not perform quite as well as the human expert, but it did outperform existing algorithmic models on the data set. To be fair, Estor would almost certainly fail to accurately estimate projects from very different environments (e.g., embedded military systems) without additional domain knowledge. Given the underlying theoretical foundations for the fundamental process of analogical reasoning in uncertain domains, the case-based approach taken by Estor should be an appropriate one; modification of Estor would be of its domain knowledge and not of the fundamental mechanisms.

An additional and perhaps more significant advantage of using the case-based approach to cost estimation is that, because it is derived from a model of human reasoning, it is intuitive and cognitively consistent with a general form of deliberation. Thus, it should be relatively easy for development managers to understand and even modify, if necessary. This feature, along with a level of accuracy that is at least equivalent and probably higher than existing models, may lead to greater acceptance of a case-based model among practitioners.

Future research needs to be directed at three areas: domain knowledge improvement, model validation, and learning. The selection of the appropriate case from memory is a crucial component of the system's domain knowledge. The
current heuristic is simplistic and might be improved by two methods. First, further study of how experts retrieve analog software projects (i.e., categorize) is needed to help understand this complex process in humans (e.g., Rips, 1989). Second, empirical studies of differential system performance with various selection heuristics would provide useful data regarding optimal selection heuristic strategies for computer-based estimation. For example, evidence suggests that the selection problem should be addressed with domain-specific knowledge (Barletta and Mark, 1988).

Recent research on software sizing metrics is also relevant to the future work in this area. Software size, it has been suggested, can be represented by a small number of factors relating to program inputs and outputs (Wrigley and Dexter, 1991). Researchers have also developed sizing methods based on application features for early estimation in the process control domain (Mukhopadhyay and Kekre, 1992). These sizing factors could be useful attributes for Estor's selection heuristic. The availability of sizing attributes whose values can be determined at an early phase in the development cycle would improve the utility of Estor as a practical estimation tool.

Finally, to be a complete cognitive model, Estor must improve its performance with the knowledge of results. Unlike existing algorithmic models, which cannot learn from successive estimation runs, human experts are able to integrate the results of observing the development project into memory and make it available for future estimation. As noted, the current model is therefore being extended to incorporate this aspect of reasoning, reflecting recent success in explicating the nature of analogical learning in humans and machines (e.g., Converse, et al., 1989; Gentner 1989).

In summary, our study has demonstrated that human expertise provides a powerful vehicle for software effort estimation. Due to its case-based orientation, Estor was able to capture and promote the intuition and deliberation necessary for software estimation. Although equipped with a limited knowledge base, Estor performed at a high level of accuracy and consistency. These results call for additional research to craft a conclusive case-based model for this difficult yet critical task faced by software development managers.

Acknowledgements

We would like to thank Dr. Herbert A. Simon for his input to this research and Dr. Chris F. Kemerer for providing us with the project data for this study. We are also grateful to the senior editor, associate editor, and anonymous reviewers for providing valuable suggestions to improve the quality of this article. This research was supported in part by a grant from the National Science Foundation, Computer and Computation Research, Software Engineering Program (CCR-8921287).

References


Jeffery, D.R. "Time Sensitive Cost Models in the Commercial MIS Environment," IEEE Trans-

About the Authors
Tridas Mukhopadhyay is assistant professor of information systems in the Graduate School of Industrial Administration, Carnegie Mellon University. He received his Ph.D. in computer and information systems from the University of Michigan in 1987. His research appears in MIS Quarterly, IEEE Transactions on Software Engineering, Information Systems Research, Journal of Operations Management, Omega, and other publications. His research interests include business value of information technology, economic impacts of electronic data interchange, and software development productivity and cost analysis. His primary area of interest is in the economics of information technology.

Steven S. Vicinanza directs the Advanced Technology Group at Energy Management Associates Inc., a division of EDS. Dr. Vicinanza was a Richard King Mellon Fellow at Carnegie Mellon University, where he received his Ph.D. in information systems from the Graduate School of Industrial Administration. He received a B.A. from Cornell University and an M.S. in computer science from the Watson School of Engineering at SUNY-Binghamton.

Michael J. Prietula is associate professor of information systems in the Graduate School of Industrial Administration (GSIA) at Carnegie Mellon University. He received his Ph.D. in management information systems from the University of Minnesota. He was a member of the Man-Machine
Appendix

Statistical Analysis

The following two hypotheses were tested in this research:

Hypothesis 1

$H_0$: The expert, Estor, COCOMO, and function point estimates are equally accurate.

$H_1$: The expert estimates are most accurate, followed by those of Estor, function point, and COCOMO models.

Hypothesis 2

$H_0$: The expert, Estor, COCOMO, and function point estimates are equally consistent.

$H_1$: The expert, Estor, COCOMO, and function point estimates are not equally consistent.

A non-parametric test was considered suitable for examining the accuracy of the four methods (hypothesis 1) because the MRE data for the four methods exhibited a lack of homogeneity of variance and the distributions appeared to be skewed. This, coupled with small sample size, suggested that a non-parametric test was appropriate (Siegel and Castellan, 1988). In particular, the Page test for ordered alternatives for matched samples was performed. For each project, the four methods were ranked based on their MRE scores. The test statistic ($z_L$) was calculated using the total rank for each method.

The consistency of each method was measured by the correlation between actual effort and estimated effort for the sample of projects. To study the consistency of the methods (hypothesis 2), pair-wise comparisons of correlations were made. Applying Fisher's $r$ to $z$ transformation for each correlation coefficient (Marascuilo and Serlin, 1988), each pair of correlations was checked for equality using an appropriate statistic ($Z_p$). For four methods, six such comparisons were made.

We predicted that the accuracy of the expert would be the highest, followed by Estor, function points, and COCOMO. Consistent with our expectations, the Page test rejected the null hypothesis that all four methods were equally accurate ($z_L = 5.09$, $p < .001$). To account for the possibility that the high MRE figures of COCOMO might be driving the result, COCOMO was dropped and the analysis was performed for the remaining three methods. The alternative hypothesis ranked the expert, Estor, and function points in decreasing order of accuracy. The null hypothesis was rejected again ($p < .05$). Finally, a Wilcoxon signed rank test (for the comparison of two methods) also rejected the hypothesis that Estor and the expert had the same level of accuracy against the alternative that the expert performed better ($p < .001$). As expected, the test results were also true for the original 10 projects.

Applying an $r$ to $z$ transformation, the correlation values in Table 3 were compared. The results indicated that there was no difference between the referent expert and Estor ($z_R = .98$, ns) and no difference between COCOMO and function points ($z_R = .55$, ns). However, both the expert and Estor had significantly higher correlation than either COCOMO or function points ($p < .05$). Thus, the expert and Estor are equally consistent, and both are more consistent than either the function point or COCOMO method. The test results were similar for the original 10 projects.
Copyright of MIS Quarterly is the property of MIS Quarterly & The Society for Information Management and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.