1990

CASE-BASED REASONING IN SOFrWARE EFFORT ESTIMATION

Steven Vicinanza
Carnegie Mellon University

Michael J. Prietula
Carnegie Mellon University

Tridas Mukhopadhyay
Carnegie Mellon University

CASE-BASED REASONING IN SOFTWARE EFFORT ESTIMATION

Steven Vicinanza
Michael J. Prietula
Tridas Mukhopadhyay
Graduate School of Industrial Administration
Carnegie Mellon University

ABSTRACT

A case-based analogical reasoning model, called Estor, was proposed and elaborated from verbal protocols gathered in a prior study. Estor incorporates five analogical problem solving processes: problem representation, analog retrieval, solution transfer, attribute mapping, and no-correspondence adjustment. These five generic processes were supplemented with the domain-specific knowledge of the referent expert. The resulting system was then presented with fifteen software effort estimation tasks, ten of which were among those solved by the referent expert, plus five new tasks. For comparison, the expert was asked to estimate the five new tasks as well. The estimates of Estor were then compared to those of the expert as well as those of the Function Point and COCOMO estimations of the projects. Significant between-estimator differences were found, with the human expert and Estor dominating the effects. Correlations between the actual effort values and the estimates of the expert and Estor for all fifteen projects were .98 and .97 respectively. Furthermore, these coefficients differed significantly from those of COCOMO and Function Points. Differences between the model and the referent expert are discussed.

1. 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), impels 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 costs (i.e., estimation of effort) to develop them. Projects originally thought to be valuable may be subsequently judged improvident. Up to 15 percent of new development projects are abandoned mid-stream, largely due to cost overruns (Jones 1986). Underestimated projects that do reach completion are often released prematurely to meet the budget, omitting important features or system testing, and resulting in systems that are incomplete and unreliable (Kemerer 1989).

Project overestimation creates problems as well. 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 programmer's time) as well as their effectiveness (e.g., by making more appropriate cost/benefit project decisions). As software becomes increasingly expensive and critical to today's organizations, the consequences of mis-estimation 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. Although a wide number of such approaches have been generated, 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 attention, their use in industry is often restricted to the verification of estimates generated by manual techniques (Zelkowitz et al. 1984).

We propose that this type of (analytical) approach is not necessarily wrong, but that it is insufficient. 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 have successfully adapted to the demands of the estimation task – the experts.

Previous research (Vicinanza, Mukhopadhyay and Prietula 1990) has suggested that expertise at estimating software effort does exist and accurate estimates can be generated by highly experienced software development managers. The most accurate of the experts studied relied upon a distinct form of reasoning to solve the estimation problems. In particular, that 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 the current study is to construct a computational model demonstrating the analogical reasoning strategy used by the most accurate human estimator. Furthermore, this study explores the model's practical utility as a technique for estimating software effort.

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 the reasoning model, called Estor, which instantiates the approach, and report on a test of Estor in which the accuracy of Estor's estimates are compared to those of Function Points, COCOMO, and the expert from whom the model was derived, on a common set of estimation problems. We conclude with a discussion of the implications to effort estimation and future research.

2. ESTIMATING SOFTWARE COSTS

Boehm (1981) describes a number of different cost estimation methods such as algorithmic models, expert judgement, analogy, and the traditional bottom-up approach, which is, perhaps, the method most widely practiced. Bottom-up estimating involves successively decomposing the development project into unit tasks until each unit or component of the project can be estimated by the individual responsible for the component's implementation. The familiarity of the project members with the components which they estimate leads to a high degree of accuracy in individual unit-task estimates. Unit-task costs can then be totalled to produce a final cost estimate. The major drawbacks of this approach include the tendency to neglect of system level costs and incidental activities such as reading, reviewing, meeting, and training. Moreover, a unit-task analysis cannot be performed in the absence of a sufficiently detailed software design. With the design of large software systems accounting for up to 40 percent of the total development cost (Conte, Dunsmore and Shen 1986, p. 6), it is often unreasonable to delay overall cost estimates until design completion.

An alternative method of estimating costs is the use of one or more algorithmic models. Cote, Boruque, Oligny, and Rivard (1988) have identified over twenty such software effort models in the literature including COCOMO (Boehm 1984), Doty (Herd et al. 1977), SLIM (Putnam 1978), PRICE-S (Frieman and Park 1979), ESTIMACS (Rubin 1983), and Function Points (Albrecht and Gaffney 1983). In general, algorithmic effort models use a combination of software size metrics and productivity factors to produce an estimate of the effort required to complete the project. The most common size metric used as input to these models is lines of code (LOC); however, other, more easily estimated size metrics such as Function Points, have also been used.

These models have the advantages of being objective (unbiased with respect to factors that are not parameters) and reliable (given the same inputs they always produce the same outputs). Although most require an estimate of software size such as LOC as input, they do not explicitly require a functional decomposition of the system and some, such as the Function Point model, have parameters that can be taken from a detailed requirements specification and do not require an LOC estimate.

Attempts have been made to seek independent validation for some of the algorithmic models. In general, these studies confirm that the accuracy of estimates generated by *uncalibrated* algorithmic models on independent project data sets is relatively low. Model calibration requires a sizable historical project database, which may not be available. A more fundamental problem with many quantitative models is that the estimation process is based on mathematical formulae whose parameters can be difficult for development managers to understand and manipulate. Consider the Function Point model, in which the total number of function points is calculated as a weighted combination of program size attributes (e.g., the number of external files). The weights assigned to each attribute have no inherent meaning to software managers - they are seemingly arbitrary multipliers which must be blindly applied to the project data. This lack of coherence between the actual task environment and the model may reduce managers' faith in the model's accuracy as well as their ability to manipulate the model to reflect the idiosyncrasies of their particular development environments. Given the problems with existing quantitative models, it is not surprising that their practical use is limited as a supplement to other methods (Zelkowitz et al. 1984).

Most alternative methods (see Boehm 1984) require the use of human expertise in one form or another to estimate development cost. Although the use of expert judgement is commonplace in industry (Wrigley and Dexter 1987), few researchers have generated empirical evidence on this approach. In the most direct examination of this method to date, Vicinanza, Mukhopadhyay, and Prietula (1990) report that experts given input parameters to COCOMO and Function Point models estimate effort with an average error as low as 32 percent. On the same project set, the Function Point and COCOMO models estimate with mean errors of 106 percent and 758 percent, respectively. Furthermore, the experts proved to be much more sensitive to factors affecting productivity than either COCOMO or Function Points, as evidenced by a markedly higher correlation between the experts' estimates and actual project efforts. Analogical reasoning was observed to be one strategy which was used to solve the estimation problem, and which produced accurate estimates. We have therefore focused our efforts on understanding and modeling this problem solving method.

3. MECHANISMS OF ANALOGY AND CASE-BASED REASONING

Analogical reasoning is a fundamental tool in the human problem solving repertoire (Sternberg 1977). As such, various aspects of this method have been studied in a wide variety of different task domains including natural language comprehension (Carbonell 1982), scientific discovery (Thagard and Holyoak 1985), dispute mediation (Kolodner, Simpson and Sycara-Cyranski 1985), accounting and tax problems (Marchant et al. 1989a), law (Marchant et al. 1989b), and business planning (Sullivan and Yates 1988). Within the context of software, Boehm (1984) recognizes it as a useful technique and Silverman (1985) identifies analogy as the primary method NASA systems designers use to estimate the size and execution time of new ground-based satellite control systems. Allen (1990) has applied case-based reasoning to the development of knowledge-bases describing orbital trajectory simulation systems. The empirical literature on analogical reasoning in software estimation is, however, virtually nonexistent.

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.

A more specific framework for studying analogical problem solving has been proposed by researchers building computational models of case-based reasoning (e.g., Kolodner, Simpson and Sycara-Cyranski 1985). 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 solution. The problem to be solved is referred to as the *target* of the analogy. The previous problem is called the *source* of the 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.

Whereas a general theory of analogical problem solving must accommodate the mapping of a source analog whose elements are syntactically remote from those of the target (i.e., across task domains), research into case-based reasoning has focused on a more common situation, where the source analog is drawn from the same general problem domain and has the same syntactic structure as the target. By constraining the source of the analogy to previously solved 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, a domain where the syntactic organization of the target case, a software project, is similar to that of previous cases, other software projects.

In case-based reasoning (e.g., Kolodner 1987), 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 fairly straightforward, as a common set of features is shared among all cases. Then the solution that achieved the goal in the source problem is transferred to the target and subsequently modified to compensate for analogical elements whose mappings are not in correspondence.

The case-based approach to problem solving is appropriate 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 in terms of task adaptation, the structure of the task is always changing and the appropriate knowledge adaptations cannot reflect deep causal principles, rather, those structures permit effective access to the most appropriate, similar experiences encountered and need not rely on underlying causal mechanisms (Prietula, Feltovich and Marchak 1989). As 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, the case-based approach evidenced by the expert should indeed be an appropriate strategy to bring to bear.

4. ESTOR: A CASE-BASED REASONING MODEL

In a prior study (Vicinanza, Mukhopadhyay and Prietula 1990), problem solving data was collected from highly experienced software managers, each individually estimating the effort required to complete each of ten software development projects. The software projects used in the study came from Kemerer (1987) and comprised 37 project factors and the actual development effort associated with each of the ten completed projects.

The projects represented medium to large data processing systems, ranged in size from 39,000 to 450,000 LOC, and required from 23 to 1,107 man-months of effort to complete. The COCOMO and Function Point inputs for these projects were converted into a suitable presentation format and used as stimulus materials.

Based on the magnitude of relative error, MRE (Conte, Dunsmore and Shen 1986) and the coefficient of determination (Albrecht and Gaffney 1983) measures of estimate to memory structures and their values can be placed, manipulated, and accessed by the inference engine.

Rule 1.

IF

Staff Size of Selected Base Project is small
AND
Staff Size of Target Project is large

THEN

Increase the Effort Estimation of the Target Project by 20 percent

RULE 2.

IF

Complexity of Target Project is two units > Selected Base Project

THEN

Increase the Effort Estimation by adding back an among equal to the Base Estimate

Figure 1. Example Adjustment Rules

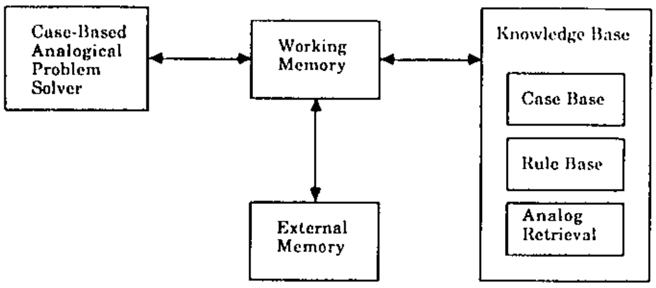

Figure 2. Overall Architecture for Estor

Figure 3 illustrates the contents of working, long-term, and external memory as a project is being estimated. External memory contains information relating to the target projects and the current estimation session. Working memory has been elaborated with pointers to information needed by the current analogical reasoning process. This example depicts a snapshot of the system just before the conflict set of adjustment rules is to be created. At this instant, a target project has been identified, a source has been selected, and non-corresponding attributes have been identified. Pointers to these structures are referenced via working memory.

5. A COMPARISON OF PERFORMANCES

5.1 Method and Procedure

Estor provided estimates for the ten estimation problems described in the prior section as well as for five new projects of the same form. In addition, estimates of the five new problems were also obtained from a COCOMO and Function Point analysis (Kemerer 1987). Finally, the referent expert for the construction of Estor also solved the five new estimation problems. It should be noted that although Estor's domain knowledge was constructed from an analysis of the protocol for the referent expert, Estor was not otherwise based on any of the initial ten problems to which the expert was responding (i.e., the ten problems were not used as "training cases").

5.2 Results and Discussion

For all estimators, two primary data were used in the analysis: final estimates and derived MREs (magnitude of relative error[2]). The first analysis examined Estor's performance estimating the original ten projects. An analysis incorporating post-hoc Tukey tests was performed comparing Estor's mean MRE scores with the mean MRE scores obtained by the referent expert, COCOMO and Function Points.

The results indicated that Estor performed better than COCOMO ($p<.05$), but not significantly better than the other estimators. A regression analysis was performed fitting a linear model to the relationship between the actual effort and estimated effort (Actual = $\alpha + \beta$ * Estimate). This analysis revealed an r^2 for ($r^2=.95$) Estor equivalent to the r^2 of the referent expert and exceeding those of either COCOMO ($r^2=.70$) or Function Points ($r^2=.62$) for the same data (see Table 1).

Estimator	R-Square	α	β	Prob (2-tailed)
Expert	.95	-25.9	1.35	<.001
Estor	.95	19.1	1.00	<.001
COCOMO	.70	-14.0	.16	<.01
Function Points	.60	-38.4	1.08	<.01

Table 1. Regression Fit for Initial Ten Cases

The overall architecture of the system is illustrated in Figure 2. The knowledge base, or long-term memory, contains the domain-specific knowledge (case base, rule base, analog retrieval heuristic). The five reasoning processes (construct, retrieve, map, transfer, adjust) are implemented by the case-based analogical problem solver or inference engine. Information is transferred via a conceptual working memory into which symbolic references to memory structures and their values can be placed, manipulated, and accessed by the inference engine.

The second analysis focused on Estor's performance on the five new cases along with the referent expert, COCOMO and Function Points. Although there were not enough cases for a regression analysis, a Kruskal-Wallis one-way ANOVA by ranks indicated that differences in MRE scores did exist ($KW=10.98$, $p<.05$). Alpha-adjusted multiple comparisons demonstrated that the referent expert had lower MREs than other estimators ($p<.05$) and that Estor and Function Points were lower than COCOMO ($p<.05$) but not significantly different themselves.

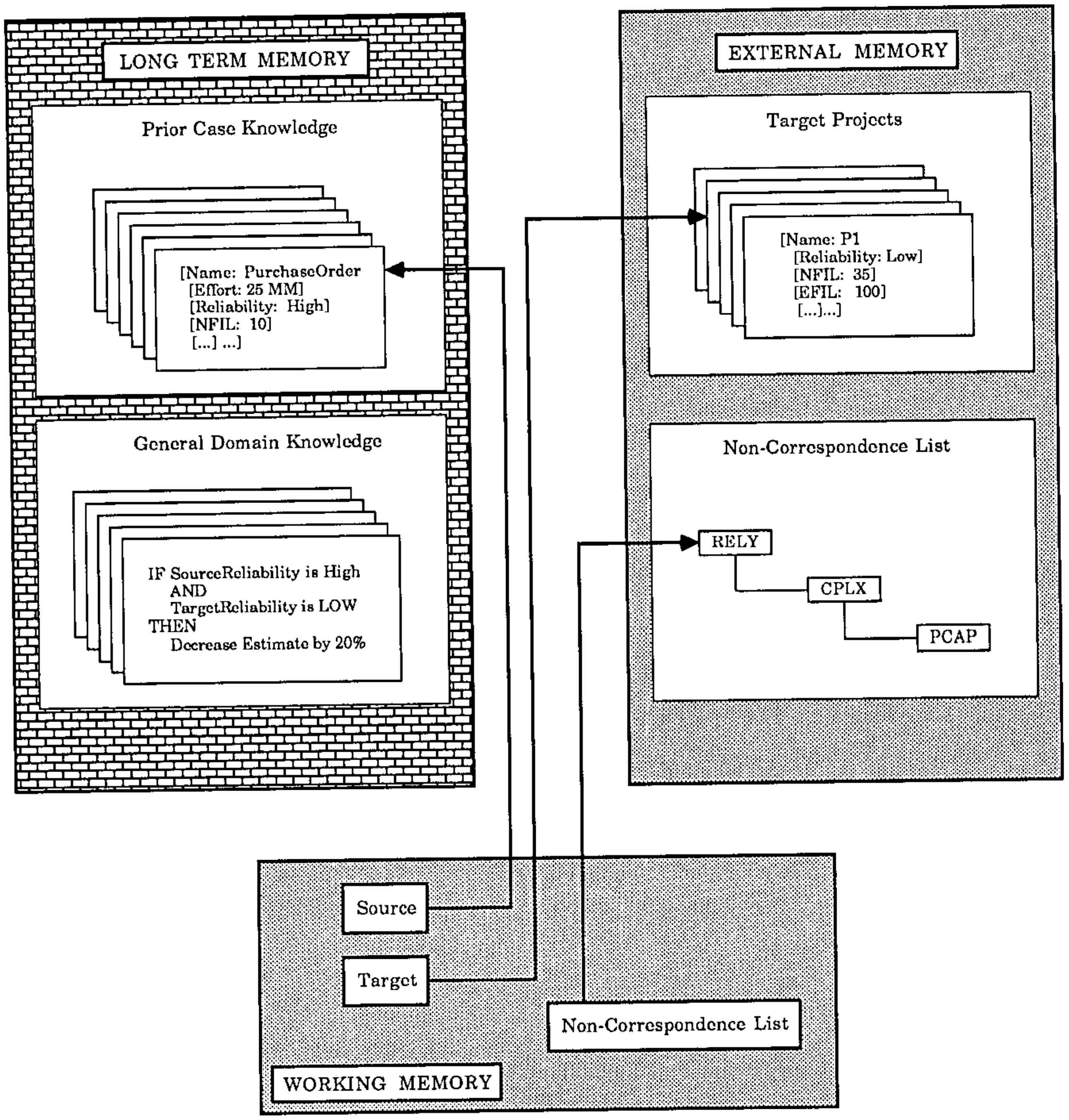

Figure 3. Illustration of Memory Contents During Estimation

The third analysis pooled all of the data from the fifteen projects. An analysis of the MRE data yielded an overall estimator effect using the Kruskal-Wallis one-way ANOVA ($KW=33.22$, $p<.001$). Alpha-adjusted multiple comparisons determined that the relationships among estimator MRE scores were quite similar to those obtained in the analysis of the five new cases – the referent expert had lower MREs than the other estimators ($p<.05$) and that Estor and Function Points were lower than COCOMO ($p<.05$). However, in this analysis Estor did have lower MRE scores than Function Points ($p<.05$). Table 2 summarizes the results of the regression analysis for this data set. Note the variance explained in both Estor's and the referent expert's r^2 as well as similar adjustments to the intercept (plausibly establishing an anchor at the origin) and coefficients ($\beta_{Estor}=1.04$, $\beta_{Expert}=1.32$). Applying an r to z transformation (Sachs 1984), we compared the correlation coefficients noted in Table 2. The results indicated that there was no difference between the referent expert and Estor (z = .98, ns) and no difference between COCOMO and Function Points (z = .55, ns). However, both the expert and Estor had significantly higher correlations than either COCOMO or Function Points (Estor > COCOMO, z = 2.23, p<.05).

Estimator	R-Square	α	β	Prob (2-tailed)
Expert	.97	--	1.32	<.001
Estor	.94	--	1.04	<.001
COCOMO	.68	22.8	.15	<.001
Function Points	.55	-36.9	.96	<.01

Table 2. Regression Fit for All Fifteen Cases

A final, qualitative analysis begins to address the 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 is 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 was previously noted, the verbal protocols contained little 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. Consequentially, 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 is in the selection of adjustment rules. Estor sometimes applied rules to mapping non-correspondences that were missed by the expert, often 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 fact, there were instances where no rules could be applied to map source to target. In these cases, Estor's estimates were much less accurate than those of the referent expert.

The identification of these points of divergence is important as it shows where the system can be modified to improve its performance. From the initial analysis, three basic improvements are a larger (though representative) case-base, a better selection heuristic, and a larger rule base to handle adjustments. These three represent the domain specific knowledge in the system. The basic analogical reasoning processes realized in the case-based reasoning model appear to be plausible and adequate to the task; furthermore, the specific addition of particular domain knowledge should significantly enhance performance.

6. CONCLUSION

This study has presented a model of case-based analogical software cost estimation and has described an instantiation 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 employed by an expert 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 this data set. To be fair, it 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, so that modification of Estor would be through its domain knowledge and not the fundamental mechanisms. To test this proposition, we are adjusting Estor to address two additional and diverse environments: embedded military software (ADA) and traditional data-processing maintenance projects.

As the sample size for the analyses presented is limited, it is our intent that the validation study be viewed as an indicator of plausibility rather than an unequivocal verification of generalizability. Studies to validate the more general applicability of this approach are in progress.

Future research needs to be directed at three areas: domain knowledge improvement, model validation, and extension. 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). We are in the process of performing such analyses. 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. The current model is therefore being extended to incorporate this aspect of reasoning reflecting recent interest in analogical learning in humans and machines (e.g., Converse, Hammond and Marks 1989; Gentner 1989).

7. ACKNOWLEDGEMENTS

We thank Chris F. Kemerer for providing us with the project data for this study, the anonymous reviewers for their helpful critiques, and Herbert A. Simon for his insightful comments and suggestions.

8. REFERENCES

Abdel-Hamid, T., and Madnick, S. "Impact of Schedule Estimation on Software Project Behavior." *IEEE Software*, July, 1986.

Albrecht A. J., and Gaffney J. "Software Function, Source Lines of Code, and Development Effort Prediction." *IEEE Transactions on Software Engineering*, SE9(6) 1983, pp. 639-648.

Allen, B. "Case-Based Reasoning in a Software Information System." Inference Corporation, Los Angeles, California. Manuscript submitted for publication, 1990.

Anderson, J. R. *The Architecture of Cognition*. Cambridge, Massachusetts: Harvard University Press, 1983.

Ashley, and Rissland, E. "Compare and Contrast: A Test of Expertise." In *Proceedings of AAAI-87*, 1987.

Barletta, R., and Mark, W. "Explanation-Based Indexing of Cases." In J. Kolodner (ed.), *Proceedings of a Workshop on Case-Based Reasoning*, Information Science and Technology Office, DARPA, 1988.

Boehm, B. W. "Improving Software Productivity" *IEEE Computer*, September, 1987, pp. 43-57.

Boehm, B. W. *Software Engineering Economics*. Englewood Cliffs, New Jersey: Prentice-Hall, 1981.

Boehm, B. W. "Software Engineering Economics." *IEEE Transactions on Software Engineering*, SE10, 1984, pp. 4-21.

Carbonell, J. "Metaphor: An Inescapable Phenomenon in Natural Language Comprehension." In W. Lehnert and M. Ringle (eds.), *Strategies in Natural Language Processing*, Hillsdale, New Jersey: Lawrence Erlbaum, 1982.

Converse, T.; Hammond, K.; and Marks, M. "Learning Modification Rules from Expectation Failure." *Proceedings of a Workshop on Case-Based Reasoning*, Information Science and Technology Office, DARPA, 1989.

Cote, V.; Bourque, P.; Oligny, S.; and Rivard, N. "Software Metrics: An Overview of Recent Results." *The Journal of Systems and Software*, 8, 1988, pp. 121-31.

Conte, S.; Dunsmore, H.; and Shen, V. *Software Engineering Metrics and Models*. Menlo Park, California: Benjamin/Cummings, 1986.

Davis, R., and King, J. "An Overview of Production Systems." In E. Elcock and D. Michie (eds.), *Machine Intelligence 8*. Cichester, England: Ellis Horwood, 1977.

Ericsson, K. A., and Simon, H. A. *Protocol Analysis: Verbal Reports as Data*. Cambridge, Massachusetts: MIT Press, 1984.

Forgy, C. "OPS5 User's Manual." Technical Report CMU-CS-81-135, Department of Computer Science, Carnegie Mellon University, 1981.

Frieman, F. R., and Park, R. D. "PRICE Software Model - Version 3: An Overview." In *Proceedings IEEE PINY Workshop on Quantitative Software Models*, 1979.

Gentner, D. "The Mechanisms of Analogical Learning." In S. Vosniadou and A. Ortony (eds.), *Similarity and Analogical Reasoning*. Cambridge, England: Cambridge University Press, 1989.

Herd, J. R.; Postak, J.; Russel, W.; and Stewart, K. "Software Cost Estimation Study - Study Results." Final Technical Report RADC-TR-77-220, Volume 1, Doty Assoc., Rockville, Maryland, 1977.

Hunter, L. "Case-Based Planning: An Integrated Theory of Planning, Learning, and Memory." Unpublished Dissertation, Department of Computer Science, Yale University, 1989.

Jones, C. "The Productivity Report Card." *Software News*, Volume 6, Number 9, 1986, p. 19.

Kemerer, C. F. "An Empirical Validation of Software Cost Estimation Models." *Communications of the ACM*, Volume 30, Number 5, 1987, pp. 416-429.

Kemerer, C. F. "An Agenda for Research in the Managerial Evaluation of Computer-Aided Software Engineering (CASE) Tool Impacts." *Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences*, IEEE Computer Society Press, 1989.

Kolodner, J. "Extending Problem Solving Capabilities Through Case-Based Inference." *Proceedings of the Fourth International Machine Learning Workshop*, 1987.

Kolodner, J. "Judging Which is the Best Case for a Case-Based Reasoner." *Proceedings of a Workshop on Case-based Reasoning*, Information Science and Technology Office, DARPA, 1989.

Kolodner, J.; Simpson, R.; and Sycara-Cyransky, K. "A Process Model of Case-Based Reasoning in Problem Solving." *Proceedings of IJCAI-85*, 1985.

Marchant, G.; Robinson, J.; Anderson, U.; and Schadewald, M. "A Cognitive Model of Tax Problem Solving." *Advances in Taxation*, Volume 2, 1989a pp. 1-20.

Marchant, G.; Robinson, J.; Anderson, U.; and Schadewald, M. "Analogical Transfer and Expertise in Legal Reasoning." Forthcoming in *Organizational Behavior and Human Decision Processes*, 1989b.

Martin, C. "Direct Memory Access Parsing." Unpublished Dissertation, Department of Computer Science, Yale University, 1989.

Minsky, M. "A Framework for Representing Knowledge." In P. Winston (ed.), *The Psychology of Computer Vision*. New York: McGraw-Hill, 1975.

Newell, A., and Simon, H. *Human Problem Solving*. Englewood-Cliffs, New Jersey: Prentice-Hall, 1972.

Prietula, M.; Feltovich, P.; and Marchak, F. "A Heuristic Framework for Assessing Factors Influencing Knowledge Acquisition." *Proceedings of the Twenty-Second Hawaii International Conference on Systems Science*, IEEE Computer Society Press, 1989.

Putnam, L. H. "A General Empirical Solution to the Macro Software Sizing and Estimating Problem." *IEEE Transactions on Software Engineering*, SE4(4), 1978, pp. 345-361.

Rips, L. "Similarity, Typicality, and Categorization." In S. Vosniadou and A. Ortony (eds.), *Similarity and Analogical Reasoning*. Cambridge, England: Cambridge University Press, 1989.

Rubin, H. A. "Macroestimation of Software Development Parameters: The ESTIMACS System." *IEEE SOFTFAIR Conference on Software Development Tools, Techniques and Alternatives*, 1983.

Sachs, L. *Applied Statistics*. New York: Springer-Verlag, 1984.

Shank, R. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge, England: Cambridge University Press, 1982.

Silverman, B. "The Use of Analogs in the Innovation Process: A Software Engineering Protocol Analysis." *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(1), 1985, pp. 30-44.

Sternberg, R. "Component Processes in Analogical Reasoning." *Psychological Review*, Volume 84, Number 4, 1977, pp. 353-378.

Sullivan, C., and Yates, C. "Reasoning by Analogy - A Tool for Business Planning." *Sloan Management Review*, Volume 29, Number 3, 1988, pp. 55-61.

Thagard, P., and Holyoak, K. "Discovering the Wave Theory of Sound: Inductive Inference in the Context of Problem Solving." *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*. Palo Alto, California: Kaufmann, 1985.

Vicinanza, S.; Mukhopadhyay, T.; and Prietula, M. "Software Effort Estimation: A Study of Expert Performance." Manuscript submitted for publication, 1990.

Wrigley, C. D., and Dexter, A. S. "Software Development Estimation Models: A Review and Critique." *Proceedings of the ASAC Conference*, University of Toronto, 1987.

Zelkowitz, M.; Yeh, R.; Hamlet, R.; Gannon, J.; and Basili, V. "Software Engineering Practices in the US and Japan." *IEEE Computer*, June, 1984.

9. ENDNOTES

1. Estor is implemented in Smalltalk/V from Digitalk, Inc.

2. MRE is calculated as 100 x |actual - estimate|/actual.