Human Judgement in Effort Estimation of Software Projects

Magne Jørgensen, Geir Kirkebøen, Dag I. K. Sjøberg, Bente Anda and Lars Bratthall
{magnej, geirki, dagsj, bentea, lbr}@ifi.uio.no
Industrial Systems Development, Department of Informatics
University of Oslo

Abstract
This paper summarises earlier attempts to improve the effort estimation in software projects and discusses what we should learn from these attempts. Based on that discussion we recommend more research on how to combine estimation models and human judgement. In order to combine estimation models and human judgement we should identify the most useful findings and methods from earlier empirical research on human judgement. We give a few examples on potential useful findings and evaluates the use of a human judgement analysis model, the Lens Model. Finally, we describe a recently started, Lens Model based, empirical study on human judgement in software estimation.

Keywords: Software effort estimation, human judgement, Lens Model

1. The Problem

Organisations developing software have, in general, a bad reputation for effort overrun. According to a recent survey carried out by Standish Group[1] more than 60% of the software projects had more than 20% effort overrun. The situation is, probably, even worse since large projects seem to be more frequently prone to large effort overruns [28] and cancelled projects were not counted. The consequences of effort overruns are, among others:

- lack of quality of the deliverables, see [26]
- dissatisfied customers
- frustrated developers

The characteristics of software projects, such as "one of a kind" activities, dynamic environments, changing requirements and carried out by humans, mean that we cannot expect zero effort overruns. On the other hand, there seems to be an agreement that software effort estimation has a large improvement potential. In this paper we argue that more research on how to combine formal estimation models and human judgement should be carried out. As a consequence, it will be important to identify:

Which findings and methods from the empirical research on human judgement are useful in order to study and to improve effort estimation in software projects?

There are, of course, other disciplines than human judgement that are important for the study and improvement of software effort estimation, such as organisational and learning theory. These disciplines are not discussed in this paper.

2. Previous Work

2.1 Effort Estimation Models

There has been a lot of work on developing formal effort estimation models; se for example [1, 5, 7, 14, 24, 33, 35]. Unfortunately, most evaluation studies [11, 18, 27, 31] indicate that the present estimation models are very inaccurate when evaluated on historical software projects different from the projects on which the estimation models were derived. In fact, the accuracy of the estimation models seems to be even worse than the situation reported by Standish Group (see previous section).

[1] http://www.standishgroup.com/chaos.html. Describes a study based on a survey of US companies.

2.2 Expert Estimation

The term "expert estimation" is not clearly defined and covers a wide range of estimation approaches. A common characteristic is, however, that "intuitive" processes constitute a major part of the estimation. Consequently, it is difficult to articulate the estimation process and to achieve organisational learning. We have found only few studies aiming at a better understanding of the expert estimation process in a software context. The results described in [10] indicate that expert estimates become more accurate when including risk analysis in the estimation process. [34] found that the estimators did not show much understanding of the cause-effect complexity of the software project environment and that the initial effort estimate had a large impact of re-estimations, i.e. a strong conservatism. [25] reported that there were no clear correlation between length of experience and prediction accuracy of own work among software maintainers. [19] found a close relationship between software tasks characteristics, such as size and type of modules subject to the changes, and expert estimation accuracy. Studies from other domains indicate several interesting characteristics of expert judgement relevant for software effort estimation. For example:

- Experts performed better than models in a highly predictable environment, but worse in a less predictable environment [21].
- Experts were able to learn from previous estimates when getting feedback on the task relations, e.g. feedback on the relation between task characteristics and actual effort. Feedback only on the outcome, such as feedback on the judgement accuracy, did not lead to learning [2].
- Experts systematically had too high confidence in their judgements, if not proper learning was achieved [16].
- Decomposition of a task for estimation purposes could activate too much information processing and lead the expert estimator astray [30].
- Experts outperformed models in shorter-term business forecasting, whereas models outperformed experts in longer-term forecasting [6].

There may be many very specific research consequences from the results described above. For example, it would be interesting to study whether the finding in [21] is valid in a software development environment. If expert estimates are better than models in highly predictable environments, but worse in less predictable environments, the consequence may be that the development of estimation models should focus on less predictable environments. This is, we believe, the opposite of what been the case in software estimation research, so far. An example: Ultra large software development projects constitutes a less predictable environment, at least for organisations that normally carry out small and medium large projects. We do not know of any research on how to support the estimation of ultra large projects with estimation models.

2.3 Combination of Models and Experts

A few studies, such as [8, 23, 28, 31], indicate that the combination of effort estimation models and expert judgement leads to improved effort estimates. Similar results are found in other domains, such as in business forecasting [4]. A potential reason for these findings are that experts and models have different strengths and weaknesses, which are useful for combination purposes. Potential strengths of a model are, for example, that the model will have less bias towards over-optimistic effort estimates [23] and that the model's effort estimate is less impacted by organisational and social pressure. Potential strengths of an expert are that the expert can identify new variables relevant for the actual project effort estimate and can include "broken-leg" situations, i.e. very rare situations that are not meaningful to include in an estimation model. While we know a lot about the characteristics of estimation models we do not have much empirical based knowledge about expert judgement. The main consequence of the findings is, in our opinion, that we should focus more on how to combine expert judgement and formal models in software effort estimation. To do this we need proper research methods and interpretation models. The "Lens Model" described in the next section of this paper is a candidate for such a model.

3. The Lens Model Applied to Software Effort Estimation

3.1 The Lens Model

The Lens Model was introduced by Brunswick [9] and adopted to a human judgement context by Hammond [20]. Since then the model has been used in many human judgement studies, see [12] for an overview of different uses. As far as we know, the only use of the Lens Model in studies on software effort estimation is the study design described in [29]. That study was, however, never carried out.

An essential feature of the Lens Model for our purpose is that expert effort estimation accuracy can be modelled and analysed as a function of:

- The predictability of the environment, i.e., the degree of effort variance explained by a formal estimation model. An estimation environment can, for example, be a set of estimations of software projects with similar characteristics regarding size and complexity.
- The "control" of the expert in the environment, i.e., the similarity between an expert's judgements and predictions based on a model of the expert's judgements.
- The "knowledge" of the expert about the environment. The Lens Model divides this knowledge into "linear" knowledge (G) and "unmodeled" knowledge (C). The linear knowledge can, given that the estimation models described above are based on linear regression, be interpreted as the expert estimator's ability to correctly ensure correspondence between his or her judgement policy and the "optimal" model. The unmodeled knowledge is the degree of variance not explained by the estimation models, for example non-linearity or missing variables in the estimation models.

3.2 Elements of a Lens Model Study of Expert Effort Estimation

To illustrate our use of Lens Model features, and the Lens Model itself, we describe elements of an empirical study of software effort estimation we have started at a large software company in Norway. This study is expected to be completed Autumn 2000. Figure 1 depicts the components of the Lens Model described in the design. The overall goal of that study is to better understand strengths and weaknesses of expert estimation.

1) Identify the indicators (X_i) used by the expert estimators to estimate the effort. Analyse the relative importance of these indicators and how they are used. An indicator may, for example, be the number of user screens that must be developed. We plan to identify and analyse these indicators through interviews with the estimators and through observation of the expert estimation process using think-aloud protocols [17].
2) Collect the expert estimated effort (Y_s) and the actual effort (Y_e) of each software project. Calculate the estimation accuracy of the expert estimates (r_a). This estimation accuracy can, for example, be calculated as the mean magnitude of relative error (MMRE), see [11], between estimated and actual effort.
3) Develop a prediction model (M1) of the actual effort (Y_e) that uses the values of the indicators (X_i) as input. The predicted effort from this model is in the Lens Model termed Y'_e.
4) Develop a prediction model (M2) of the expert's effort estimates (Y_s) that uses the values of the indicators (X_i) as input. This is a model of the expert's "estimation policy". Note the difference between the actual expert effort estimate (Y_s) and the predicted expert effort estimate (Y'_s). The quality of the model of the expert's "estimation policy" is important for our analysis. If this model, maybe adjusted for personal characteristics, explains major parts of the expert judgements, we may gain insight in when we should expect accurate estimates from an expert and when a model of type M1 is better. Currently, we do not know how dependent on personal characteristics this "estimation policy" will be. For this reason, we collect information about the experience, education, subjective evaluation of own estimation skills and cognitive style.
5) Calculate the predictability of the environment (R_e). The predictability can, for example, be calculated as the MMRE (see step 2) between actual effort and effort predicted by M1.
6) Calculate the control of the expert estimates (R_s). The control can, for example, be calculated as the MMRE (see Step 2) between expert estimated effort and effort predicted by M2.

7) Analyse the knowledge of the expert estimators (G + C). This analysis can, for example, be based on an analysis of the similarity of the estimation models M1 and M2, the use of intercorrelated indicators, see [36], and an analysis of the omission of important indicators in M1 or M2.

These steps enable an analysis of the differences between formal estimation models an expert estimates, both in terms of accuracy, control and knowledge, in different environments. The knowledge elicited from that analysis, we believe, will be important for more insight in the strengths and weaknesses of the expert estimators. In particular, we expect to see differences in the estimation accuracy, control and knowledge of the expert estimator due to the types of projects to be estimated. We plan to compare the distribution properties (variance, skewness and kurtosis) of the variables as well as the accuracy of the predictions. The quality of the M2-model

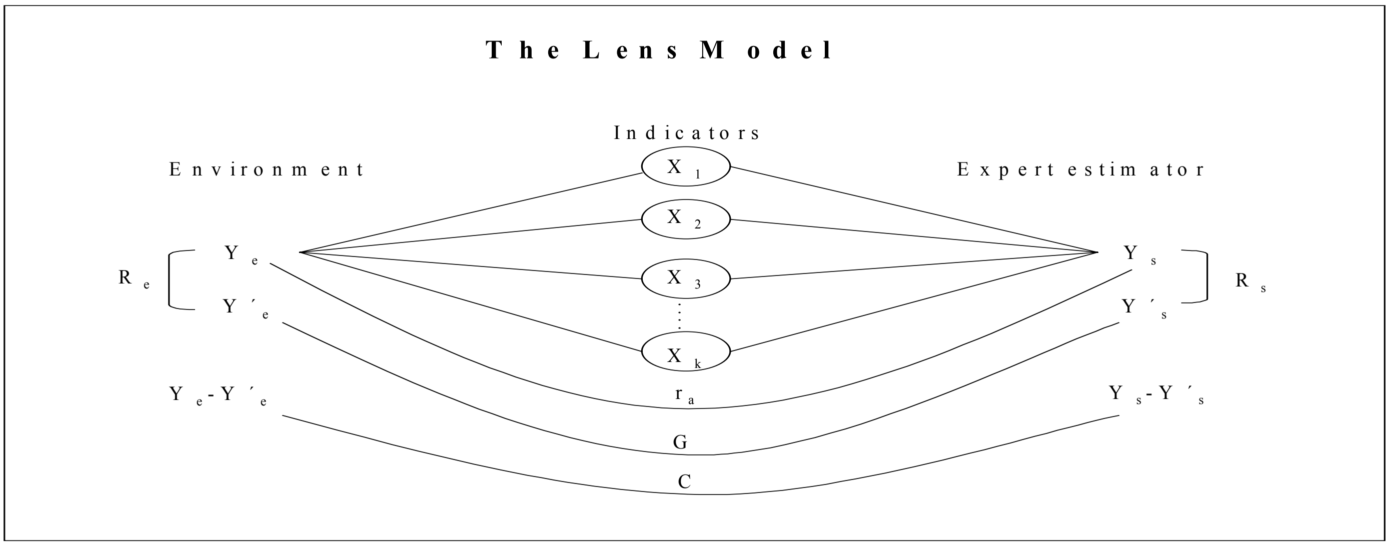

Figure 1 The Lens Model

3.3 The Lens Model Equation

If we base our estimation models on linear multiple regression and the estimation accuracy is measured as the multiple linear correlation, the relation between the components of the Lens Model can be described more formally. The Lens Model Equation can be formulated as:

$$r_a = R_e R_s G + C((1-R_e^2)(1-R_s^2))^{1/2}$$

with the variables interpreted as in Section 3.1. Furthermore, if we assume that the unmodelled component equals zero (C=0), we have that the estimation accuracy is the product of the predictability of the environment, the control of the expert estimator and the knowledge of the expert estimators.

3.4 Evaluation of the Lens Model

There are advantages and disadvantages connected with the use of the Lens Model in our context.

Potential advantages:

- The Lens Model is a well-established framework and a lot of study designs and results can easily be borrowed from and compared with other human judgement studies.
- The Lens Model gives an interpretation of important terms and introduces important distinctions not made in current software effort estimation studies, such as the distinction between predictability of the environment, the expert's control and knowledge.

Potential disadvantages:

- The Lens Model seems very much based on multiple regression analysis and correlation based measures of estimation accuracy, i.e., linear relations. Typically, software estimation models and studies are based on non-linear relations and other measures of estimation accuracy. This means that the Lens Model Equation may not be very useful.

- Pattern recognition (estimation by analogy) may be an important “mental” strategy when experts estimate software effort. The Lens Model seems to assume that the subject’s policy must be based on a number of indicators, which may be different from a pattern recognition model.
- The Lens Model does not give very much explicit support for analysing “intuition”, i.e. the process of deriving the judgement policy.
- There is no simple way of handling nominal scale variables in the Lens Model.
- It would be no surprise if the real estimation policy of an expert estimator was based on intercorrelated indicators, i.e., that there is a redundancy in the information used by the expert. The quality of the evaluation of the expert’s control depends on insight in the type and nature of this incercorrelation. It may be difficult to gain this insight from the observations of and the interviews with the expert estimators.

There may be ways to get around these potential disadvantages. Non-linear relationships may be covered using log-linear regression models, other prediction accuracy measures may be used in addition to the correlation measure, the pattern recognition component may be considered as one out of many indicators in the Lens Model and nominal scale variables can be introduced using logit transformations. Cooksey discusses possible solutions to some of these problems in more detail [12]. The performance of linear models is important for the use of the Lens Model Equation. As discussed in [13], linear models seem to have the highest success in environments where:

1. There is a monotone relation between each of the indicators and the actual effort.
2. The relation between an indicator and the actual effort is independent of the values of the remaining indicators.
3. There is an error of measurement of the indicators. As the error increases the optimal model will, according to [13], be more linear.
4. An error in the measurement of the actual effort does not affect the model of the relative importance of the indicators.

In which degree these conditions are met in a software effort estimation environment depends on the indicators that are included in the estimation models. However, the major input to most current estimation models, the size of the software to be developed, seems to meet conditions 1 and 3. There is a monotone relation between the size and the effort and there is, typically, a large degree of error in the measurement of this indicator.

Interestingly, most current software effort estimation models are developed and evaluated on historical project data where the size of the software is measured without errors, i.e., the size of the software is measured when the project is completed. These estimation models are, typically, non-linear. In order to carry out a fair comparison between linear and non-linear effort estimation models, one should compare them using indicator values with a realistic level of measurement error.

To evaluate the linearity of a typical effort estimation model we carried out a comparison of the performance of a non-linear regression based effort estimation models of the type: Effort = a Size^b with a linear regression model of the type Effort = a + b Size based on the Albrecht-Gaffney-dataset [1]. First, we assumed no measurement error and developed and evaluated (using cross-validation) a linear and a non-linear estimation model with “Function Points” as Size-variable. The b-value of the non-linear models was significantly ($p<0.001$) higher than 1 and the MMRE (prediction accuracy) of the non-linear model was significantly better than the MMRE of the linear model (one-sided paired t-test, $p = 0.01$). Then, we simulated the performance of the estimation models assuming measurement error of the input values. For example, if the actual software size of a project (measured at completion) was 100 Function Points, the possible size input to the model would be picked randomly from a beta-distribution with mean value 100 and a max value of 130 a min value of 70 (with alpha=beta=2). With what we believed was realistic measurement error (10-20% mean measurement error on the size and effort input values) the linear model now resulted in a (non-significant) better predictions (lower MMRE)! Consequently, the non-linearity of the software effort estimation environment may, not be as obvious as it may look from the frequency of non-linear estimation models in the literature. A more analytical way to analyse the impact of measurement error is described in, for example, [32].

We need to use the Lens Model in real studies of software effort estimation to decide on its usefulness. The study briefly described in Section 3.2 has at the time of completion of this paper reached the data collection phase. We expect preliminary results to be available in May/June 2000.

4. Final Comments

Studies of expert judgement in the effort estimation process are, in our opinion, strongly underrepresented. A major reason for only finding a few studies on this topic may be that it is difficult for software engineering researchers to navigate in the large number of research methods, results and theories in the psychology and human judgement discipline. In addition to the Lens Model there are a lot of alternative or complementary models to study of human judgement, for example:

- The Signal Detection Theory [15]
- The Image Theory [3]
- The Conflict Theory [22]

In our opinion, software estimation researchers need to co-operate with psychologists to base their work on existing knowledge on human judgement and to choose the most appropriate research and interpretation model. Only this way we can avoid "reinventing the wheel".

Acknowledgements
We gratefully acknowledge the support from our industrial partner, IconMedialab, in particular Alexander Woxen and Per Einar Dybvik. The research project is funded by The Research Council of Norway through the industry-project PROFIT (PROcess improvement For the IT industry).

References:

[1] A. J. Albrecht and J. E. Gaffney, "Software function, source lines of code, and development effort prediction," *IEEE Transactions on Software Engineering*, vol. 9, 639-648, 1983.

[2] W. K. Balzer, M. E. Doherty, and R. J. |O'Connor, "Effects of cognitive feedback on performance," *Psychological Bulletin*, vol. 106, 410-433, 1989.

[3] L. R. Beach and T. R. Mitchell, "Image theory: Principles, goals and plans," *Acta Psychologica*, vol. 66, 201-220, 1987.

[4] R. C. Blattberg and S. J. Hoch, "Database models and managerial intuition: 50% model + 50% manager," *Management Science*, vol. 36, 887-899, 1990.

[5] B. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[6] P. A. Braun and I. Yaniv, "A case study of expert judgment: Economists' probabilities versus base-rate model forecasts," *Journal of behavioral decision making*, vol. 5, 217-231, 1992.

[7] L. C. Briand, V. R. Basili, and W. M. Thomas, "A pattern recognition approach for software engineering data analysis," *IEEE Transactions on Software Engineering*, vol. 18, 931-942, 1992.

[8] L. C. Briand, K. El Emam, and F. Bomarius, "COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking, and Risk Assessment," *Int. Conf. on Software Engineering*, Kyoto, Japan, 1998; 390-399.

[9] E. Brunswik, "Organismic achievement and environmental probability," *Psychological Review*, vol. 50, 255-272, 1943.

[10] T. Conolly and D. Dean, "Decomposed versus holistic estimates of effort required for software writing tasks," *Management Science*, vol. 43, 1029-1045, 1997.

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

[12] R. W. Cooksey, *Judgement analysis. Theory, methods and applications.* Calefornia: Academic Press, 1995.

[13] R. M. Dawes and B. Corrigan, "Linear models in decision making," *Psychological Bulletin*, vol. 81, 95-106, 1974.

[14] W. A. Delaney, "Predicting the costs of computer programs," *Data processing magazine*, 32-34, 1966.

[15] J. P. Egan, *Signal detection theory and ROC analysis*. New York: Academic Press, 1975.

[16] H. J. Einhorn, "Confidence in judgment: Persistence of the illusion of validity," *Psychological review*, vol. 85, 395-416, 1978.

[17] K. A. Ericsson and H. A. Simin, *Protocol analysis: Verbal reports as data (Revise Ed.).* Cambridgde, MA: The MIT Press, 1993.

[18] G. R. Finnie and G. E. Wittig, "A comparison of software effort estimation techniques: using function points with neural networks, case based reasoning and regression models.," *J. Systems Software*, vol. 39, 281-289, 1997.

[19] A. R. Gray, S. G. MacDonell, and M. J. Shepperd, "Factors systematically associated with errors in subjective estimates of software development effort: the stability of expert judgment," *Proceedings Sixth International Software Metrics Symposium*, 1999.

[20] K. R. Hammond, "Probabilistic functionalism and the clinical method," *Psychological Review*, vol. 62, 255-262, 1955.

[21] S. J. Hoch and D. A. Schkade, "A psychological approach to decision support systems," *Management Science*, vol. 42, 51-64, 1996.

[22] I. L. Janis and L. Mann, *Decision making: a psychological analysis of conflict, choice, and commitment*. New York: The Free Press, 1977.

[23] M. Jørgensen, "An empirical evaluation of the MkII FPA estimation model.," *Norwegian Informatic Conference*, 1997; 7-18.

[24] M. Jørgensen, "Experience with the accuracy of software maintenance task effort prediction models," *IEEE Transactions of Software Engineering*, vol. 21, 674-681, 1995.

[25] M. Jørgensen, D. Sjøberg, and G. Kirkebøen, "The Prediction Ability of Experienced Software Maintainers," *Accepted for publication: 4th European conference on software maintenance and reengineering*, Zürich, Switzerland, 2000; .

[26] M. Jørgensen and D. I. K. Sjøberg, "Empirical studies on effort estimation in software development projects," *IRMA 2000 (accepted for publication)*, Alaska, 2000; .

[27] C. F. Kemerer, "An empirical validation of software cost estimation models," *Communications of the ACM*, vol. 30, 416-429, 1987.

[28] B. Kitchenham, S. L. Pfleeger, B. McColl, and S. Eagan, "A case study of maintenance estimation accuracy," *Submitted to IEEE Transactions on software engineering*.

[29] C. R. Litecky, "Cost estimation in programming using a lens model of program complexity," *The National Conference Proceedings of the Decision Science Institute*, Miami, Florida, 1991; .

[30] D. G. MacGregor and S. Lichtenstein, "Problem structuring aids for quantitative estimation," *Journal of behavioral decision making*, vol. 4, 101-116, 1991.

[31] I. Myrtveit and E. Stensrud, "A controlled experiment to assess the benefits of estimating with analogy and regression models," *IEEE Transaction of Software Engineering*, vol. 25, 510-525, 1999.

[32] R. S. Pindyck, *Econometric models and economic forecasts*: McGrawHill, 1997.

[33] L. H. Putnam, "A general empirical solution to the macro software sizing and estimation problem," *IEEE Transaction on Software Engineering*, vol. 4, 345-381, 1978.

[34] K. Sengupta and T. K. Abdel-Hamid, "The impact of unreliable information on the management of software projects: a dynamic decision perspective," *IEEE Transaction on systems, man, and cybernetics*, vol. 26, 177-189, 1996.

[35] M. Shepperd and C. Shofield, "Estimating software project effort using analogies," *IEEE Transactions on Software Engineering*, vol. 21, 126-137, 1997.

[36] T. R. Stewart, "Judgment analysis: Procedures," in *Human judgment: The SJT view*, B. Brehmer and C. R. B. Joyce, Eds. Amsterdam: North-Holland Elsevier, 1988, 41-74.