Viewpoints

Software Development Effort Estimation: Formal Models or Expert Judgment?

Magne Jørgensen and Barry Boehm

Which is better for estimating software project resources: formal models, as instantiated in estimation tools, or expert judgment? Two luminaries, Magne Jørgensen and Barry Boehm, debate this question here. Outside this article, they're colleagues with a strong inclination to combine methods. But for this debate, they're taking opposite sides and trying to help software project managers figure out when, and under what conditions, each method would be best.

While it might be less argumentative—and certainly less controversial—to agree that the use of both methods might be best, Magne is making a stronger point: Perhaps our refinement of and reliance on the formal models that we find in tools is wrong headed, and we should allocate our scarce resources toward judgment-based methods. (For a primer on both methods, see the sidebar.) — Stan Rifkin

//Layout artist: Place the sidebar near here.//

Magne: Formal software development effort estimation models have been around for more than 40 years. They're the subject of more than a thousand research studies and experience reports. They're described and promoted in many software engineering textbooks and guidelines. They're supported by user-friendly tools and advisory services from consultancy companies. In spite of this massive effort and promotion, available empirical evidence shows that formal estimation models aren't in much use.[1] Formal effort estimation models have had more than sufficient opportunities to become a success story; it's now time to move on and focus industrial estimation process improvement work and scientific research on judgment-based effort estimation methods. This is a much more promising way forward for several reasons. Here are three.

First, 10 out of the 16 studies reviewed in "Estimation of Software Development Work Effort: Evidence on Expert Judgment and Formal Models" report that using (typically simple and unstructured) judgment-based effort estimation methods led to more accurate effort estimates than using sophisticated formal models.[2] The introduction of more structure and more supporting elements, such as structured group processes and experience-based estimation checklists, could even further improve judgment-based effort estimates. See, for example, *Principles of Forecasting: A Handbook for Researchers and Practitioners* for general forecasting evidence on the benefits of several types of expert-judgment process structures and support.[3]

Second, so far, little work has been done on improving judgment-based effort estimation processes compared with that on formal effort estimation models. As an illustration, Martin Shepperd and I found that only 15 percent of the journal papers on software development effort estimation analyzed judgment-based effort estimation.[4] Unlike most papers on formal effort estimation models, the great majority of these papers didn't aim at improving judgment-based effort estimation processes.

Third, most of the software industry, as far as I've experienced, is much more willing to accept, understand, and properly use judgment-based estimation methods. I have, for example, repeatedly observed that software projects officially applying a formal estimation model actually use the model as a disguise for expert estimation. This low acceptance and "models as judgment in disguise" attitude appear in several forecasting disciplines and shouldn't be surprising. An important reason for the rejection of formal models might be that experts' highly specific knowledge—for example, about the developers who are supposed to do the work—frequently can't be included properly as model input. It's understandably difficult to trust an estimation method unable to make use of strongly relevant information. It's hardly possible to unite highly specific knowledge with the need to establish general relationships in formal models; that is, this limitation isn't a question of improved models.

In short, there are very good reasons to claim that future estimation process improvement and research initiatives should aim at better judgment-based effort estimation processes and not at better formal models.

Although formal effort estimation models might seem primarily objective and repeatable, I believe they aren't. All meaningful estimation models require judgment to produce the input to the models. This might

include judgment of complexity, team skill, and the customers' requirements for the system. This means that software companies benefit from a stronger focus on better judgment-based processes—even when they choose to apply estimation models, for example, in combination with expert judgment.

Barry: First, let me say that I think Magne's published analyses are well done and important to understand. However, I don't think that a 10/16 = 62.5 superiority percentage for expert judgment in empirical studies is conclusive, particularly because there are uncertainties in whether the results of empirical studies are representative of results in project practice. If most people are using expert judgment methods in practice, the results aren't encouraging. The last few years of Standish Group surveys indicate that only about 30 percent of the projects surveyed successfully deliver their software within their estimates.[5] T. Capers Jones' analysis of 50 completed projects indicated that only four (8 percent) were estimated within 10 percent of their actuals.[6] However, these data do support Magne's call for more research into better methods of judgment-based estimation.

One candidate explanation of this discrepancy is that the expert estimates produced in empirical studies aren't representative of estimates produced in practice. Some nonrepresentativeness occurs because not everyone is an expert, nobody is an expert in all domains, and empirical studies based on estimates and data from completed projects aren't representative of estimates made when they're needed, at the beginning of projects. As an educator, I wonder what the definition of "expert" is; when exactly does someone go from being a nonexpert to an expert, and, perhaps more important, what can we do to accelerate that process? I'm not sure about the definition of "expert," and substituting a synonym such as "experienced" doesn't help, because we would again have to face the question of exactly when someone goes from being inexperienced to experienced. And, of course, I'm wondering whether the expertise to which Magne refers is something that can be both taught and learned, or something that can only be learned.

A well-documented problem with the accuracy of estimates, particularly early in the life cycle, is the Cone of Uncertainty, which indicates that early estimates might be off by factors of two to four, simply because not enough is known about either the requirements or the solutions (for example, the feasibility of using off-the-shelf components).[7,8] When faced with such uncertainties, many organizations highly value the ability to perform extensive sensitivity, risk, and trade-off analyses. The number of cases involved in such analyses goes well beyond the available time of experts to perform, whereas parametric models can run them rapidly, with little human effort.

When comparing expert-judgment versus parametric-model estimates, another factor might cause parametric models to appear less accurate. The organization's counting rules used for software size, cost, and schedule might differ from those used to calibrate the models. For size, this might involve differences between logical and physical lines of code, counting of nonexecutable statements, and counting of support software such as deliverable tools and test drivers. For cost and schedule, it might involve differing phases counted (programming versus full development cycle), work hours counted (training, personal activities, and overtime), and activities counted (configuration management, quality assurance, data preparation, and hardware and software integration). For Cocomo, my colleagues and I found factor-of-three differences between an evaluation study's data and data conforming to Cocomo's published definitions.

In situations involving high cost and schedule uncertainty, it's a good idea to draw upon as many sources of insight as possible. Parametric models contain a good deal of information about which factors cause software costs to go up or down, and by how much. This information is based on extensive analyses of real-project data and feedback from real-project users. Their absolute estimates might differ from your actuals owing to the causes I just mentioned, but their relative estimates of the effects of various cost drivers are generally fairly accurate. As a bottom line, I agree with Magne's recommendation for more research on better methods of expert-judgment estimation. But I would strongly disagree with any recommendations to stop people from performing research on which factors affect software costs and by how much, or to discourage people from using insights provided by parametric models.

Magne: Barry and I agree that the accuracy of judgment-based effort estimation isn't exactly amazing. Judgment-based estimates are far from perfect. For example, they frequently involve a high degree of wishful thinking and inconsistency. The surprising observation is therefore that estimation models haven't managed to produce more accurate effort estimates! This observation is even more surprising when you consider that models seem to outperform expert judgment in most other disciplines.[9]

Barry suggests that poor model accuracy can be a result of size-counting rules that differ from those

used to calibrate the model. I share his concern about the lack of standardized size measurement and agree that there's an improvement potential related to the practical use of models. His suggestion does, however, also nicely illustrate how complex and time consuming it can be to use effort estimation models properly. An organization must therefore decide whether it should prioritize its limited resources on training and monitoring people's size-counting processes, or on better methods and support for judgment-based effort estimation.

Estimation models have inherent problems that make the latter choice more worthwhile and that explain the empirical results in favor of the expert judgment. In particular, the relation between effort and size in software development contexts isn't stable. The results reported in "On the Problem of the Software Cost Function,"[10] based on an evaluation of 12 data sets, illustrates how much the functional relation between effort and size vary from data set to data set. This lack of relation stability is a problem not only between data sets but also within data from the same organization. Essential effort relationships might, for example, change over time owing to learning or technology change. In situations with unstable relationships, expert judgment has been shown to be preferable.[11]

In addition, as mentioned before, software development situations frequently contain highly specific, highly important information. Take, for example, the information that there's a serious conflict between the project leader Peter and his best programmer Paul. This conflict will likely have a serious impact on the project's productivity. No effort estimation model, as far as I know, integrates this kind of relation mechanically. A model might choose to integrate this type of highly specific information through expert judgment—for example, through expert judgment of how much such conflicts will likely impact overall productivity. Then, however, dominant model variables would be judgment-based and subject to biases and problems similar to those of a purely expert-judgment-based process. Consequently, it isn't easy to see how the models would be able to perform much better than experts in such situations. Several studies—for example, "On Knowing When to Switch from Quantitative to Judgemental Forecasts"[12]—suggest that expert judgment can have great advantages in situations with highly specific information that's not mechanically integrated, or integrated at all, in the model.

Even if models should tend to be less accurate than expert judgment, software organizations might, according to Barry, benefit from using them as a means to sensitivity, risk, and trade-off analyses. I'm not so sure, and I've seen no empirical evidence to support the usefulness of this. Simulation of how a single variable affects effort may require an ability to model all the underlying relationships that's even better than that needed for effort estimation. That's why simulation models typically are much more complex than estimation models.

The representativeness of the 16 empirical studies comparing estimation models and judgment-based effort estimation might not be optimal. And, of course, these studies don't prove that judgment-based estimation models are on average better. These studies do, on the other hand, provide the best available knowledge about this issue and should therefore be important input when deciding where to put estimation improvement effort. I hope, however, that there soon will be more studies shedding light on *when* models and judgment-based estimation processes are likely to be more accurate.[3]

Barry: I'll try to respond to Magne's main points in his second-round contribution.

First, Magne argues that properly using estimation models requires more investment in data collection, model calibration, education, and training. I agree. It correlates with a recent Chinese survey of 116 organizations involved in process improvement, in which this investment was the highest-ranked (52 percent) of the "barriers and difficulties to applying cost estimation models."

Often, though, organizations find these investments worthwhile. Recently, my colleagues and I administered a survey to 16 acquisition managers at a software acquisition cost and risk workshop and to 28 cost estimation professionals at a cost estimation forum. They answered several questions of the form "How much value do current parametric cost models contribute when you perform the following functions, as compared to human-judgment estimates?" on a scale of 1 (much less) through 3 (about equal) to 5 (much more).

The top-ranked relative values of parametric cost models (with the average responses for the two groups) were

- preparing cost proposals (4.3, 4.4),
- evaluating cost proposals (3.75, 4.0),

- performing risk or sensitivity analyses (3.6, 3.8), and
- making cost-schedule-performance trade-offs (3.45, 3.8).

The top-ranked relative values of human-judgment estimates were

- performing make-buy-outsource analyses (2.35, 2.7),
- investing in productivity improvements (2.35, 2.8),
- allocating project budgets (2.5, 3.05), and
- making software evolution or phaseout decisions (2.7, 2.9).

You can see not only that the results vary with the population being sampled but also that the results vary even more strongly by the cost estimation function you're trying to perform. So, at least in these sampled populations, making a one-size-fits-all decision on using models versus experts in all situations doesn't appear to be a good idea.

Second, Magne states that

> *The relation between effort and size in software development contexts isn't stable.*

Again, I would agree, and go further in saying that this phenomenon has been well known for quite a while and is the main reason that parametric cost models have incorporated additional productivity-driver parameters to explain the variation. Chapter 29 of *Software Engineering Economics* shows the significant difference between the original Basic Cocomo model with a size parameter and a single-mode parameter (estimating the 63 projects in the database within a factor of 1.3 of the actuals only 29 percent of the time), and the Intermediate Cocomo model with an added 15 productivity-driver parameters (within a factor of 1.3, 78 percent of the time).[7]

Several purported evaluations of the accuracy of "Cocomo" didn't have cost driver data and could only evaluate Basic Cocomo, in which case it wasn't surprising that "Cocomo" didn't produce very accurate results.

Third, Magne says that

> *Software development situations frequently contain highly specific, highly important information [such as personnel compatibility].*

Such information is often incorporated in the leading parametric models. For example, both the Cocomo family and SEER[13] productivity-driver parameters Analyst Capability and Programmer Capability rate team rather than average individual capability, including such factors as raw ability, efficiency and thoroughness, and ability to communicate and cooperate. It's true, though, that parametric models never have enough calibration data to accurately estimate the effect on the productivity of 99th-percentile teams. It's also true that well-calibrated parametric models are biased toward the results of organizations that collect and are willing to share their productivity data. These tend to be better-performing organizations, although highly capable agile teams can be even more productive without needing to collect and analyze data.

Finally, according to Magne,

> *Judgment-based estimates tend to have a higher degree of wishful thinking.*

I agree. A major advantage of a parametric model is that it doesn't modify its estimates when customers, managers, or marketers apply pressure.

The only way you can get a parametric model to reduce its estimates is by going on record to make a visible change in the project's estimated size or in its productivity-driver parameter ratings. Thus, using a calibrated parametric model enables negotiation of the price of a software development contract to be driven by objective adjustment of project size or productivity-driver parameters, rather than by a contest of wills between self-described experts.

Barry: Because Magne has been gracious enough to start first in Rounds 1 and 2 and let me have the last word in each round, and because I think we've covered a lot of the significant ground in those rounds, I propose that we each make closing statements, that I'll go first, and that he can have the last word.

Rather than continue to discuss which is better of two estimation methods that I think are both complementary and useful, I'd like to focus on what you should do with the estimates that you've produced or received. Unfortunately, for the many organizations that consistently overrun, the usual practice is to discard them as having served their purpose and to avoid future embarrassment when the estimates are overrun. However, by doing so, such organizations lose any chance of learning through experience. They also remain vulnerable to overly optimistic expert- or model-based estimators.

The organizations I've seen that have had the most success at meeting cost and schedule commitments use a mix of parametric models and expert judgment. But more important, they save the estimates and use them in a closed-loop feedback process to compare estimate inputs and outputs with actual experiences. Any mismatches not only can be corrected during the project but also can be used to calibrate the inputs for future project estimates. For example, consistently overoptimistic expert estimators or model users can be warned or excluded, domain experts can be calibrated for their optimism or pessimism about how others will succeed in their domain, or model parameters can be recalibrated to evolving experience.

I used to think that closed-loop feedback and recalibration would enable organizations and models to become increasingly perfect estimators. But I don't any more. The software field continues to reinvent and rebaseline itself too rapidly to enable this to happen. As a bottom-line statement, we need to recognize that people and models will never be perfect software cost and schedule estimators, and build this into our project plans.

A good way to do this is to use incremental development and to use the process called either *timeboxing* or *cost and schedule* as an independent variable. This involves customers prioritizing their desired features, and developers architecting their systems to make borderline-priority features easy to add or drop. Then, as each increment goes on and finds itself ahead of or behind its estimated cost and schedule, it can add or drop borderline features to make the increment fit its budget or schedule. This enables the customers to receive their most valuable features in each increment. However, it also recognizes that if estimation can't be perfect, then the number of such features can't be an independent variable along with cost and schedule. But again, it requires using the estimates to monitor progress with respect to plans, and applying closed-loop feedback processes within each increment, across increments, and across projects to keep on top of changing situations.

Magne: My main claim in this discussion is that organizations' process improvement work and research initiatives should focus on better judgment-based effort estimation processes, not on introducing or improving formal estimation models. The claim is backed up with results from a comprehensive review of relevant empirical estimation studies and explanations based on results from other forecasting disciplines. As far as I can see, none of Barry's arguments provides evidence against this claim.

Barry's arguments are instead related to the observation that some software professionals find estimation models useful, that model use can be improved, that we should combine models and expert judgment, and that more frequent feedback through incremental models improves estimates. These claims are easy to agree on but don't easily translate to an organization's decision on whether to improve model- or judgment-based estimation processes.

The current strong belief in models among some software professionals and many software engineering researchers might be a result of myths about models and expert estimation, possibly illustrated by Barry's claim that

> *use of a calibrated parametric model enables negotiation of the price of a software development contract to be driven by objective adjustment of the project's size or productivity-driver parameters, rather than by a contest of wills between self-described experts.*

The first myth here is that models are objective. As I stated earlier, they clearly aren't. Essential input to the models is based on expert judgment and can consequently be impacted by outside pressure.

The second myth is that judgment-based effort estimation is a black-box process leading to a contest of wills between self-described experts—that is, a process that's difficult to improve. This isn't true, either. There are many ways to improve judgment-based effort estimation and avoid contests of wills. Barry's own

Wideband Delphi process and the guidelines suggested in "Practical Guidelines for Expert-Judgment-Based Software Effort Estimation"[14] are examples of such ways.

A third myth, especially common among software engineering researchers, is that more advanced estimation models will likely lead to substantially more accurate effort estimates. For example, that the intermediate version of the Cocomo models or a neural-network-based estimation model will likely be more accurate than much simpler models. In software engineering, as in most other similar forecasting disciplines, the perhaps most stable result is that simple models typically perform just as good as more advanced models and that more sophisticated models are vulnerable to overfitting to outdated or unrepresentative data sets. The improvement potential of the model side will consequently likely be quite low.[1,3]

Although we clearly shouldn't reject estimation models as providing no value, we shouldn't ignore available empirical evidence, either. If the software engineering discipline wants to become evidence based, it should focus on improving those processes for which the evidence suggests that improvements will more likely occur. In most effort estimation contexts, but hardly all, the available evidence suggests a much stronger focus on improving judgment-based effort estimation processes.

Acknowledgments

We're indebted to Steve McConnell, Mike Cohn, and the anonymous reviewers for their thorough, candid feedback. We also thank Hakan Erdogmus, IEEE Software*'s editor in chief, for suggesting this format for the discussion of this controversy.*

References

1. M. Jørgensen, "A Review of Studies on Expert Estimation of Software Development Effort," *J. Systems and Software*, vol. 70, nos. 1–2, 2004, pp. 37–60.
2. M. Jørgensen, "Estimation of Software Development Work Effort: Evidence on Expert Judgment and Formal Models," *Int'l J. Forecasting*, vol. 23, no. 3, 2007, pp. 449–462.
3. J.S. Armstrong, ed., *Principles of Forecasting: A Handbook for Researchers and Practitioners*, Kluwer Academic Publishers, 2001.
4. M. Jørgensen and M. Shepperd, "A Systematic Review of Software Development Cost Estimation Studies," *IEEE Trans. Software Eng.*, vol. 33, no. 1, 2007, pp. 33–53.
5. J. Johnson, *My Life Is Failure*, Standish Group Int'l, 2006.
6. T.C. Jones, *Estimating Software Costs*, McGraw-Hill, 1998.
7. B. Boehm, *Software Engineering Economics*, Prentice Hall, 1981.
8. S. McConnell, *Software Project Survival Guide*, Microsoft Press. 1998.
9. R.M. Dawes, D. Faust, and P.E. Meehl, "Clinical versus Actuarial Judgment," *Science*, vol. 243, no. 4899, 1989, pp. 1668–1674.
10. J.J. Dolado, "On the Problem of the Software Cost Function," *Information and Software Technology*, vol. 43, no. 1, 2001, pp. 61–72.
11. R.G. Webby and M.J. O'Connor, "Judgemental and Statistical Time Series Forecasting: A Review of the Literature," *Int'l J. Forecasting*, vol. 12, no. 1, 1996, pp. 91–118.
12. N.R. Sanders and L.P. Ritzman, "On Knowing When to Switch from Quantitative to Judgemental Forecasts," *Int'l J. Operations & Production Management*, vol. 11, no. 6, 1991, pp. 27–37.
13. D. Galorath and M. Evans, *Software Sizing, Estimation, and Risk Management*, Auerbach, 2006.
14. M. Jørgensen, "Practical Guidelines for Expert-Judgment-Based Software Effort Estimation," *IEEE Software*, vol. 22, no. 3, 2005, pp. 57–63.

Magne Jørgensen *is a professor at the Simula Research Laboratory. His research focuses on improving judgment-based effort estimation. He has worked as a software developer and project manager for*

several years and trained estimators to use estimation models and expert-judgment-based estimation processes. Contact him at magnej@simula.no.

***Barry Boehm** is a professor in the University of Southern California's Computer Science and Industrial and Systems Engineering Departments. He led the development of the Cocomo and Cocomo II parametric estimation models, the Wideband Delphi method for expert-judgment estimation, the Bayesian calibration method for combining expert judgment and data analysis into a parametric estimation model, and the Agile Cocomo II method and tool for combining analogy-based and parametric-model-based estimation. Contact him at boehm@usc.edu.*

***Stan Rifkin** is the founder and principal of Master Systems, an advisory service for organizations for which computing is strategic. Contact him at sr@master-systems.com.*

The Difference between Model- and Expert-Judgment-Based Estimation

The essential difference between formal-model-based and expert-judgment-based effort estimation is the quantification step—that is, the final step that transforms the input into the effort estimate. Formal effort estimation models, such as Cocomo and function-point-based models, are based on a mechanical quantification step such as a formula. On the other hand, judgment-based estimation methods, such as work-breakdown structure-based methods, are based on a judgment-based quantification step—for example, what the expert believes is the most likely use of effort to complete an activity. Judgment-based estimation processes range from pure "gut feelings" to structured, historical data and checklist-based estimation processes. That is, there are many other ways of improving judgment-based processes than selecting the best expert.

Some estimation methods, such as the Planning Poker method (www.planningpoker.com/detail.html), can be implemented as judgment- or model-based, depending on whether the final step is mechanically based on the velocity (productivity) of previous iterations. This illustrates that the difference between models and expert judgment isn't always clear. Table 1 summarizes typical similarities and differences of the processes involved.

Table 1. A comparison of the processes for expert-based and model-based estimation.

Estimation activity	Expert judgment	Estimation models
Understand the estimation problem.	Judgment- and analysis-based processes possibly structured by templates for work breakdown.	Judgment- and analysis-based processes possibly structured by the need for model input if the model has already been chosen at this stage.
Agree on decisions and assumptions relevant for estimation.	Judgment- and analysis-based processes possibly structured by checklist or guidelines.	Judgment- and analysis-based processes possibly structured by checklists or guidelines. Several assumptions implicitly made when selecting a model.
Collect information relevant for the estimation.	Judgment- and analysis-based processes possibly structured by checklists or guidelines. The set of variables considered relevant is judgment based. The collected information includes subjective assessments.	Judgment- and analysis-based processes structured by the model's need for input data. The set of variables considered relevant is predetermined. The collected information typically includes subjective assessments.
Evaluate the importance (weighting)	Judgment-based processes.	Analysis-based processes based on

of different pieces of information.		statistical analysis of historical data.
Quantify the effort on the basis of the information.	Judgment-based processes.	Analysis-based processes based on a formalized, repeatable process or formula.
Review the effort estimate.	Judgment- and analysis-based processes possibly structured by checklists, guidelines, and expert judgment.	Judgment- and analysis-based processes possibly structured by checklists, guidelines, and expert judgment, sometimes leading to judgment-based updates of the effort estimate.