

Evidence-Based Cost Estimation for Better-Quality Software

Tim Menzies and Jairus Hihn

Evidence-based reasoning is becoming common in many fields. It's widely enshrined in the practice and teaching of medicine, law, and management, for example. Evidence-based approaches demand that, among other things, practitioners systematically track down the best evidence relating to some practice; critically



appraise that evidence for validity, impact, and applicability; and carefully document it.

One proponent of evidence-based software engineering is David Budgen of Durham University. In the Internet age, he argues, many sources of supposed knowledge—Google, Wikipedia, digg.com, and so on—surround us. At his keynote address at the 2006 Conference on Software Engineering Education and Training, Budgen asks, how should we train students to assess all that information and to separate the sense from the nonsense? In his view, before we can denounce some inaccuracy in, say, Wikipedia, we must first look to our own work and audit our own results.

Today's cost estimation practices

Sadly, little of the cost estimation literature would satisfy a proponent of evidence-based software engineering. For example, what formally documented evidence exists that the current set of cost modeling “best practices” avoids the following cost/quality death spiral?

Underestimating(budget) → cost cutting → less(quality assurance and verification and validation) → lower(quality)

This death spiral begins when projects start running over budget. Sadly, such cost overruns are common. In 2004, for example, the US National Aeronautics and Space Administration's Government Accounts Office issued the report *Lack of Disciplined Cost-Estimating Processes Hinders Effective Program Management*. This report found that NASA space mission development costs exceed initial estimates by 12.8 percent on average (with a range of -44 to 94 percent). The Jet Propulsion Laboratory's experience is similar: at the 2003 AIAA Space conference, we reported that mission software development costs grow from their original estimates by 50 percent on average (with a range of -10 to 180 percent).

While cost overruns are common, budget increases are not. Obtaining an increase can be very difficult; for example, increasing a NASA budget might require months of lobbying Congress to pass a new funding bill. So, most proj-

ects require some cost cutting. All too often, that means managers must skimp on verification and validation or quality assurance activities. For example, developers might scrap unit testing or software integration and test ("we'll just move straight to system testing") or rush a premature design into production.

To avoid the cost/quality death spiral, accurate cost estimates are vital. But making such estimates involves inherent process problems. At the 1991 International Conference on Software Engineering, we documented our cost estimation experience at the JPL: we found that although formal cost methodologies are well documented, software managers often don't use them. Two years later, we were able to explain this. Software professionals resist using formal cost estimation methods because the mental models they use and the sequence of steps they take are more closely related to a case-based reasoning approach than a regression-based model. This is further complicated by what we call the *large variance* problem. Cost data inherently have a lot of noise that makes standard regression-based models brittle or unstable. This makes software managers mistrust current regression-based cost estimation methods.

While the academic literature has proposed many costing methods, including clustering, neural networks, and case-based reasoning, cost estimation in industry centers around parametric regression-based techniques such as COCOMO, PRICE-S, SEER-SEM, and SLIM. Two common industry practices for adjusting models to local conditions are

- local calibration (LC)—adding "tuning parameters" to a model that are set using local data, and
- stratification—given a database of past projects and a current project to be estimated, restricting local calibration to records from similar projects.

Both techniques seem arguably useful. But evidence-based software engineering demands more than just saying "it worked well enough for the last 20

years." It requires some sort of systematic and critical appraisal.

A sample critical appraisal

Table 1 shows a critical appraisal of LC and stratification, in which we applied LC to data from three software engineering data sets: CocII, Coc81, and Nasa93. The CocII repository is proprietary, whereas you can download the other two from the Promise repository of SE data at <http://promise.site.uottawa.ca/SERepository/datasets-page.html>. The data contain 24 subsets, or stratifications. For each stratification, we randomly selected 10 records 30 times to be the test set. We then learned a cost model using the COCOMO LC procedure from the remaining records. We then applied the learned cost model to the remaining $i = 1..10$ test items and logged the mean magnitude of the relative error (that is, $S_i(\text{abs}(\text{predicted}_i - \text{actual}_i)/\text{actual}_i)/10$). The average-error column shows average MMRE values over the 30 trials:

- If LC on stratified data reduces the cost estimation error, then the error seen in the cost models learned from the stratifications should be less than for the error seen in the cost models learned from all the data. In table 1, this appears as a green bar that falls short of the vertical black line representing 100 percent.
- If stratification increases error, the bar crosses the line and becomes red.

Table 1 contains some evidence that stratification and LC are useful. For example, the last line shows what happens when a cost model is learned from a stratification that holds just the 20 records relating to the Sub2 stratifica-

tion. In that case, stratification and LC reduced the model's prediction error from 60 percent (when we use all the Nasa93 records) to 46 percent. In this case, and in 12 others in table 1, best practices improved costing.

However, the table also has as much evidence *against* the claim that LC on stratified data improves cost estimation. There's no change on line 2, and in 11 others (lines 1, 3, 5, 10, 12, 13, 16, 17, 18, 19, 23), stratification and local calibration increase the error, sometimes incredibly so (see line 23).

For the past 20 years, local calibration and stratification have been widely cited as cost estimation "best practices." So why is our evidence so ambiguous? Our answer is twofold.

First, table 1 shows mean performance but not variance. The variances can be very large, which complicates cost estimation research. Much of our current research focuses on this variance problem.

Second, improvements in cost estimation methods haven't been driven by formal evidence-based analysis. In support of evidence-based reasoning, we routinely place our tools and data online so that others can repeat, improve, or refute our results. Unfortunately, in the cost estimation literature, publicly available tools and data are rarely used, partly because of the data's competitively sensitive nature.




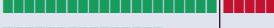
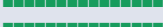
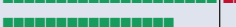
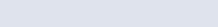


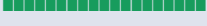
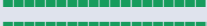



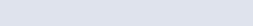
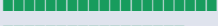
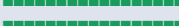
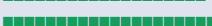





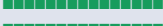
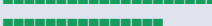

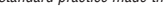
Our workbench

In a review of 17 cost estimation experiments (presented at ICSE 2005), we found numerous examples where minor details (such as how the experiments were performed) confused the authors' conclusions.

So, we built COSECKMO, a workbench for repeatable cost estimation experiments. COSECKMO can derive and evaluate numerous cost estimation tools, including LC and stratification. Currently it's designed around a COCOMO data set but could be generalized to a wider range of cost models. Experiments with COSECKMO (see table 1) show that although LC and stratification are sometimes useful, we can significantly reduce estimation mean error and variance using other methods such

To avoid the cost/quality death spiral, accurate cost estimates are vital.

Table I**Cocomo LC procedure applied to stratifications of data
from three software repositories***

Line	Source	Subset	Number of examples (a)	Mean error (b) (%)	b/a (%)	b/a = 100%
	Coc81	All	63	42	100	
1	Coc81	Kind = max	31	47	111	
2	Coc81	Mode = embedded	28	42	100	
3	Coc81	Lang = fortran	24	50	119	
4	Coc81	Mode = organic	23	32	76	
5	Coc81	Kind = min	21	47	111	
6	Coc81	Lang.mol	20	34	80	
	CocII	All	161	20	100	
7	CocII	DevelopmentEnd = 1998	54	11	55	
8	CocII	Organization = 2	48	19	95	
9	CocII	DevelopmentEnd = 1980	48	19	95	
10	CocII	Dev.waterfall	48	27	135	
11	CocII	Organization = 1	34	9	45	
12	CocII	DevelopmentEnd = 1991	22	27	135	
13	CocII	Language = C	21	23	115	
	Nasa93	All	93	60	100	
14	Nasa93	Sub5	80	53	88	
15	Nasa93	Mode = semiDetached	69	58	96	
16	Nasa93	Sub4	39	80	133	
17	Nasa93	Sub9	38	81	135	
18	Nasa93	Sub7	38	68	113	
19	Nasa93	Sub8	37	82	136	
20	Nasa93	Sub3	37	43	71	
21	Nasa93	Sub1	30	43	71	
22	Nasa93	Sub6	23	56	93	
23	Nasa93	Mode = embedded	21	188	313	
24	Nasa93	Sub2	20	46	76	

*Results in green show where standard practice improved cost estimation; results in red show where standard practice made the models worse.

as model trees, full regression, or feature subset selection.

However, our work on COSECKMO is hardly enough to change experimental methods in the cost estimation community. So, we're also running a new workshop series called PROMISE devoted to repeatable software engineering experiments. Papers submitted to PROMISE should come with the data used to reach those conclusions. For more information on PROMISE, see <http://unbox.org/promise/2006>.

The techniques and tools required to address these issues are publicly available. The software community can bring to bear many methods that could further improve evidence-based cost estimation. Nothing prevents us from moving forward except bad habits. 🐛

Acknowledgment

We performed this research at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the US Na-

tional Aeronautics and Space Administration. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not constitute or imply its endorsement by the United States Government.

Tim Menzies is an associate professor in the Lane Department of Computer Science at West Virginia University. He's worked with NASA on software quality issues since 1998. Contact him at tim@menzies.us.

Jairus Hihn is a principal member of the engineering staff at the Jet Propulsion Laboratory and manager for the Software Quality Improvement Projects Measurement Estimation and Analysis Element. Contact him at jairus.hihn@jpl.nasa.gov.