Using Risk to Balance Agility and Discipline: A Quantitative Analysis

Barry Boehm
University of Southern California
Department of Computer Science
boehm@sunset.usc.edu

Keywords
agile, discipline, architecting, risk, sweet spot

We have shown several qualitative analyses [2, 3] indicating that one can balance the risks of having too little project discipline with the risks of having too much project discipline, to find a "sweet spot" operating point which minimizes the overall risk exposure for a given project. We have shown qualitatively that as a project's size and criticality increase, the sweet spot moves toward more project discipline, and vice versa.

However, these results would have stronger credibility if shown to be true for a quantitative analysis backed up by a critical mass of data. Here we show the results of such a quantitative analysis, based on the cost estimating relationships in the COCOMO II cost estimation model and its calibration to 161 diverse project data points [1]. The projects in the COCOMO II database include management information systems, electronic services, telecommunications, middleware, engineering and science, command and control, and real time process control software projects. Their sizes range from 2.6 thousand lines of code (KLOC) to 1,300 KLOC, with 13 projects below 10 KLOC and 5 projects above 1000 KLOC.

The risk-balancing analysis is based on one of the calibrated COCOMO II scale factors, "Architecture and Risk Resolution," called RESL in the COCOMO II model. Calibrating the RESL scale factor was a test of the hypothesis that proceeding into software development with inadequate architecture and risk resolution results would cause project effort to increase due to the software rework necessary to overcome the architecture deficiencies and to resolve the risks late in the development cycle – and that the rework cost increase percentage would be larger for larger projects.

The regression analysis to calibrate the RESL factor and the other 22 COCOMO II cost drivers confirmed this hypothesis with a statistically significant result. The calibration results determined that for this sample of projects, the difference between a Very Low RESL rating (corresponding to an architecting investment of 5% of the development time) and an Extra High rating (corresponding to an investment of over 40%, here established at 50%) was an extra 7.07% added to the exponent relating project effort to product size. This translates to an extra 18% effort for a small 10 KSLOC project, and an extra 91% effort for an extra-large 10,000 KSLOC project.

The full set of effects for each of the RESL rating levels and corresponding architecting investment percentages are shown in Table 1 for projects of sizes 10, 100, and 10000 KSLOC. Also shown are the corresponding total-delay-in-delivery percentages, obtained by adding the architecting investment time to the rework time, assuming a constant team size during rework to translate added effort into added schedule. Thus, in the bottom two rows of Table 1, we can see that added investments in architecture definition and risk resolution are more than repaid by savings in rework time for a 10,000 KSLOC project up to an investment of 33%, after which the total delay percentage increases.

This identifies the minimum-delay architecting investment "sweet spot" for a 10,000 KSLOC project to be around 33%. Figure 1 shows the results of Table 1 graphically. It indicates that for a 10,000 KSLOC project, the sweet spot is actually a flat region around a 37% architecting investment. For a 100 KSLOC project, the sweet spot is a flat region around 20%. For a 10 KSLOC project, the sweet spot is at around a 5% investment in architecting. The term "architecting" is taken from Rechtin's System Architecting book [5], in which it includes the overall concurrent effort involved in developing and documenting a system's operational concept, requirements, architecture, and life-cycle strategic plan. It is roughly equivalent to the agilists' term, Big Design Up Front (BDUF) [4]. Thus, the results in Table 1 and Figure 1 confirm that investments in architecting and BDUF are less valuable for small projects, but increasingly necessary as the project size increases.

Conference '00, Month 1-2, 2000, City, State.

Table 1. Effect of Architecting Investment Level on Total Project Delay

COCOMO II RESL Rating	Very Low	Low	Nominal	High	Very High	Extra High
% Architecting Investment	5	10	17	25	33	>40 (50)
Scale Factor Exponent for Rework Effort	1.0707	1.0565	1.0424	1.0283	1.0141	1.0000
10 KDSI Project: -Added Rework %	18	14	10	7	3	0
-Project Delay %	23	24	27	32	36	50
100 KDSI Project: -Added Rework %	38	30	21	14	7	0
-Project Delay %	43	40	38	39	40	50
10,000 KDSI Project: -Added Rework %	91	68	48	30	14	0
-Project Delay %	96	78	65	55	47	50

However, the values and sweet spot locations presented in Figure 1 are for nominal values of the other COCOMO II cost drivers and scale factors. Projects in different situations will find that "their mileage may vary." For example, a 10-KSLOC safety-critical (COCOMO II RELY factor rating = Very High) project will find that its sweet spot will be upwards and to the right of the nominal-case 10-KSLOC sweet spot. A 10,000-KSLOC highly-volatile (COCOMO II Requirements Volatility factor = 50%) project will find that its sweet spot will be higher and to the left of the nominal-case 10,000-KSLOC sweet spot, due to the costs of BDUF rework. Also, various other factors can affect the probability (and size) of loss associated with the RESL factor, such as staff capabilities, tool support, and technology uncertainties [1]. And these tradeoffs are only considering project delivery time and productivity and not business value, which would push the sweet spot for safety-critical projects even further to the right. Clearly, there are a number of further issues and situations deserving of additional analysis.

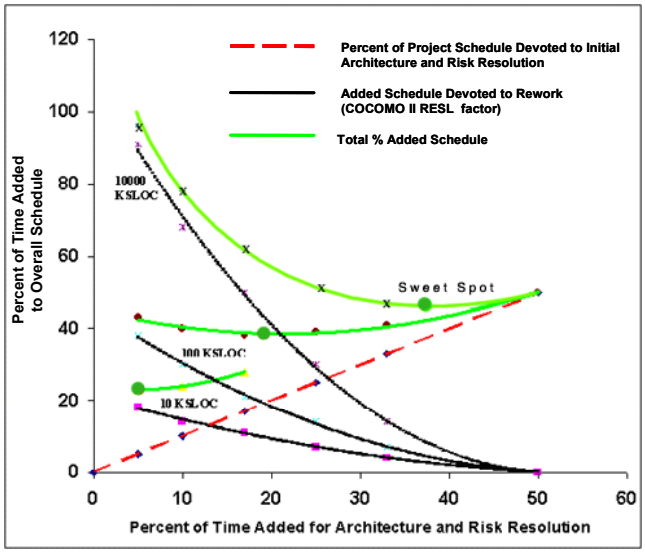

Figure 1. How Much Architecting is Enough?

REFERENCES

[1] B. Boehm, C. Abts, A.W. Brown, S. Chulani, B. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece, Software Cost Estimation with COCOMO II, Prentice Hall, 2000.

[2] B. Boehm and W. Hansen, "The Spiral Model as a Tool for Evolutionary Acquisition," CrossTalk, May 2001, pp. 4-11.

[3] B. Boehm, "Get Ready for Agile Methods, With Care," IEEE Computer, January 2002, pp. 64-69.

[4] P. McBreen, Questioning Extreme Programming, Addison Wesley, 2003.

[5] E. Rechtin, Systems Architecting, Prentice Hall, 1991.