Software Cost Estimation Meets Software Diversity

Barry W. Boehm
USC
941 Bloom Walk, SAL 328
Los Angeles, CA 90089, USA
1-213-740-8163
boehm@usc.edu

***Abstract*—The previous goal of having a one-size-fits-all software cost (and schedule) estimation model is no longer achievable. Sources of wide variation in the nature of software development and evolution processes, products, properties, and personnel (PPPPs) require a variety of estimation models and methods best fitting their situations. This Technical Briefing will provide a short history of pattern-breaking changes in software estimation methods; a summary of the sources of variation in software PPPPs and their estimation implications; a summary of the types of estimation methods being widely used or emerging; a summary of the best estimation-types for the various PPPP-types; and a process for guiding an organization's choices of estimation methods as their PPPP-types evolve.**

Keywords-Software cost; software effort; cost estimation; schedule estimation; software instructions; function points; story points; expert consensus; software diversity

I. Introduction

The Standish Group's analysis of its collection of over 25,000 software projects over the years 2011-2015 indicated an annual range of 27-31% Successful (on-budget, on-time, value-satisfactory) projects; 17-22% Failed (cancelled or rejected on delivery) projects, and 49-56% Challenged (rated satisfactory but well over budget and schedule) projects, in which the average annual budget overrun was in the 68-72% range and the average annual schedule overrun was in the 66-81% range [6]. The magnitude of the overrun percentages suggests a likely misuse or non-use of existing estimation methods and models; lack of cross-checking model inputs and outputs with expert judgement; and/or ignorance of which method or model to use in a given situation.

On the positive side, though, the significant number of successful projects suggests that better performer and manager understanding of the nature and diversity of their projects and estimation methods and models, and use of the estimates to manage the projects, has been important to their success. And for the future, it is even more timely and important to continually update and recalibrate estimation models for projects involving higher security levels, Internets of Things, massive-data analytics, self-driving cars, mixes of agile and plan-driven project elements, dynamic global supply chains, and safety-critical software-driven human prosthetics.

II. Historical Perspective

This Technical Briefing will begin with a short history of software estimation accuracy. Early software cost model developers found that as software phenomenology became better understood, the cost models became more and more accurate, leading to an expectation that they would become as accurate as the estimation of planets in their orbits. However, the increasing understanding also led to insights for increasing software productivity via software reuse, COTS products, agile methods, and cloud services, all of which required rethinking the cost models, which would then be successful until the next set of insights made them less accurate, as seen in Figure 1.

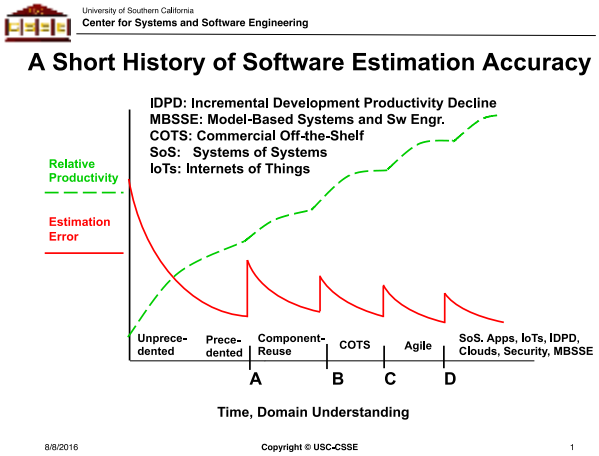

Figure 1. A Short History of Software Estimation Accuracy

Currently, the major challenges for accurate software cost estimation are not single product-oriented changes, such as component reuse, COTS, and cloud services, but a diversity of emerging product, process, property, and personnel options (PPPPs). For example, the choice of process determines the pattern of project expenditures. They can be organized into sequential phases (requirements, design, code, test) as with the waterfall on V models. They can be organized into sequential increments, as with agile methods or pre-planned product improvement combinations of sequential phases. They can be organized into continuous reprioritization of components and phases, as with Kanban and DevOps, or into incremental reprioritization of increment content as with the spiral model and Rational

978-1-5386-1589-8/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSE-C.2017.159

Unified Process. Or they can be fully concurrent and performer-prioritized as with open-source and crowd-sourcing processes.

The choice of property or quality requirements can affect both the sequencing and the amount of effort required. For example, determining or changing a system's required level of safety or security can significantly change the amount and distribution of project expenditures. Often, the best architecture and associated cost of a system is a discontinuous function of the property requirement, such as response time. On one example project, an initial requirement for a 1-second response time would have required a $100 million custom architecture, while a 4-second response time system could be developed using a COTS-based architecture for only $30 million.

The choice of project personnel has significant impacts on a project's budget. Most mainstream cost estimation models have cost-driver parameters covering personnel capability levels; domain, technology, and tools experience levels; personnel continuity levels; and team cohesion levels.

III. Estimation-Type Options

Here are the main types of software sizing and estimating techniques, with references to the best information on how to use them.

- Expert-Judgement [8]; Stakeholder Consensus
  - Planning Poker [3], Wideband Delphi [1], Bottom-Up [1]
- Analogy: Previous Projects; Yesterday's Weather [3]
  - Agile COCOMO II [2], Case-Based Reasoning [13], IDPD
- Parametric Models
  - COCOMO/COSTAR [1,2], Knowledge Plan [7], SEER [4], SLIM [11], True-S [14]
- Resource-Limited
  - Cost or Schedule as Independent Variable (CAIV, SAIV) [2]
- Reuse-Driven: Equivalent Size
  - Adjusted for %Design,Code,Test Modified, Understandability [2, 10]
- Product Line
  - % Development for Reuse; % Development with Reuse [2,10]

IV. Fitting Estimation Types to Diversity Types

Here is a mapping between diversity types and their best fits as estimation types.

- Pure Agile: Planning Poker, Agile COCOMO II
- Architected Agile
  - COSYSMO [15] for architecting; Planning Poker, CAIV-SAIV for sprints, releases; IDPD for large systems
- Formal Methods: $/SLOC by Evaluated Assurance Level
- NDI/Services-Intensive: Oracle, SAP, other ERP
  - RICE Objects: (R)eports, (I)nterfaces, (C)onversions, (E)nhancements
  - COCOTS [2], Value-Added Function Points [5,7], Agile for portions
- Hybrid Agile/Plan-Driven
  - Expert Delphi, Parametric Models, Agile for portions; IDPD
- Systems of Systems
  - COSYSMO [15] for Integrator; Hybrid Agile/Plan-Driven for component systems
- Family of Systems: COPLIMO [2, 10]
- Brownfield: Experiment for refactoring; above for rebuilding

In addition, references [9], [12], and [13] provide further valuable perspectives on estimation methods and their use in diverse situations. The Technical Briefing will close with a process for guiding an organization's choices of estimation methods as their PPPP-types evolve.

Acknowledgment

This material is based upon work supported in part by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract H98230-08-D-0171. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology. It is also supported by the National Science Foundation grant CMMI-1408909, Developing a Constructive Logic-Based Theory of Value-Based Systems Engineering.

References

[1] B. Boehm, Software Engineering Economics, Prenticce Hall, 1981.

[2] 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.

[3] M. Cohn, Agile Estimating and Planning, Prentice Hall, 2005.

[4] D. Galorath and R. Evans, Software Estimation and Risk Management, Auerbach, 2006.

[5] D. Garmus and D. Herron, Function Point Analysis, Addison Wesley, 2001.

[6] J. Johnson, My Life Is Failure, Standish Group Intl., 2016.

[7] C. Jones, Estimating Software Costs, McGraw Hill, 1998.

[8] M. Jorgensen, A review of studies on expert estimation of software development effort, J. Systems and Software 70 (2004) 37–60.

[9] S. Mc Connell, Software Estimation, Microsoft Press, 2006.

[10] J. Poulin, Measuring Software Reuse, Addison Wesley, 1996.

[11] L. Putnam and W. Myers, Measuring for Excellence,Yourdon Press, 1992.

[12] R. Stutzke, Estimating Software-Intensive Systems, Addison Wesley, 2005.

[13] A. Trendowicz and R. Jeffery, Software Project Effort Estimation, Springer, 2014.

[14] True Planning User Manual, Price Systems, 2002-2016.

[15] R. Valerdi, The Constructive Systems Engineering Cost Model (COSYSMO), VDM Verlag, 2008.