

Value-Based Software Engineering
Barry Boehm, USC
February 2002

The Future of Software Economics

The major current focus of software economics has been on the production aspects of software development at the project level. This has led to an extensive body of knowledge for software project cost estimation, software risk management, and such techniques as earned value for project monitoring and control.

In the future, this focus will be increasingly broadened in two major dimensions:

- Considerations of value and return on investment as well as cost;
- Organization-level and portfolio-level economics as well as project-level economics.

These changes are being driven by the increasing leverage that software approaches have on an organization's competitiveness and profitability, and by the critical role of software in an organization's ability to adapt to the increasingly rapid rates of change in the information technology marketplace.

The next sections address these trends from the perspective of an organization's software engineers, and the value-related challenges and opportunities confronting software engineers and their organizations.

What Are You Getting Paid For?

If you're a professional software engineer, you're getting paid good money for your efforts. What do you think that people ultimately responsible for your paycheck feel that they are paying for? Your immediate bosses may say that you're getting paid to produce designs, code, tests, and so forth. But their sources of support are expecting something different.

The ultimate sponsors of your project are expecting that the project's end result will be to add more value for them than they are paying you and the project team to create it. Their value proposition may be a financial return on investment or an improved public service like health, education, or defense. Or it might be scientific curiosity, a political objective, or pure ego satisfaction.

Why Should You Care?

It used to be that the decisions you made on a software project were pretty much decoupled from the value propositions that established the project. A requirements analysis could establish the requirements for the software, and all you were responsible for was the traceability from your software back to the requirements. But in today's world of rapidly changing information technology, organizations, and marketplaces, the "requirements" tend also to change rapidly, and in ways that require participation of all knowledgeable parties in determining just how a system's definition should change.

As a software engineer, you are both a critically knowledgeable and a critically responsible party in this new system-level process. In particular, just saying, "Oh, I'll do

whatever's needed on the same software budget and schedule," is a recipe for disaster both for your career and for your sponsor's value propositions. Thus, traceability to value propositions becomes more important and relevant than traceability to requirements. You need to be able to understand and deal with other stakeholders' value propositions and they with yours. If they want new features, they must be prepared to drop lower-priority features or add budget and schedule.

What Difference Does This Make?

A value-based approach to software engineering involves the use of new perspectives, tools, skills, and success criteria for most of the activities involved in software engineering. Let's look at some of the major activities involved: pre-requirements, requirements, architecture, development and modification, planning and control, quality assurance, people factors, and overall process organization and management.

Pre-Requirements. Software engineers need to participate in analyzing options and negotiating tradeoffs among stakeholders' often-conflicting value propositions (e.g. wanting many features and limited budget or schedule). Key skills and tools include business case analysis [Reifer, 2001], Results Chains [Thorp, 1998], business workflows, COTS evaluation, and risk assessment.

Does This Mean That We Need to Reorganize Everything?

Fortunately not. Several approaches have been evolving in this direction in response to the changes in the information technology marketplace. The Rational Unified Process is organized around the economics of software development, and has emerging extensions addressing business case analysis [Royce, 1998, p-96] and business modeling [Jacobson et al., 1994; Kruchten, 2001]. The Capability Maturity Model-Integrated (CMMI) extends the software CMM to address system-level considerations such as operational concept definition, stakeholder shared vision achievement, and risk management [SEI, 2001; Atern et al., 2001]. The spiral model's risk-driven approach has been extended into a value-driven approach called Model-Based (System) Architecting and Software Engineering (MBASE) [Boehm-Hansen, 2001; Boehm-Port, 2001], which is compatible with RUP, CMMI, and organizational approaches such as the Experience Factory [Boehm et al., 2002]. Thus, there are ways to evolve from current approaches to increasingly robust value-based approaches.