Guest Editors' Introduction: Return on Investment.

Article *in* IEEE Software · May 2004

DOI: 10.1109/MS.2004.1293068 · Source: DBLP

3 authors, including:

Hakan Erdogmus
Carnegie Mellon University
141 PUBLICATIONS **3,515** CITATIONS

SEE PROFILE

John Favaro
Trust-IT
112 PUBLICATIONS **1,881** CITATIONS

SEE PROFILE

Return on Investment

Hakan Erdogmus, *National Research Council, Canada*
John Favaro, *Consulenza Informatica, Italy*
Wolfgang Strigel, *Software Productivity Center, Canada*

If the software engineering community were to hold a contest to select its most overworked, misused, and abused acronym, it's quite possible the winner wouldn't be AI, RAD, or even OO but the unassuming little term ROI. Ubiquitous and reckless promotion by vendors, consultants, and marketing gurus has diluted the expression *return on investment* into an umbrella term that can mean anything from profits to competitive advantage to simply "something good." Consequently, the software community looks upon ROI with increasing suspicion as a vague and slippery gimmick used chiefly to make the sales pitch (invariably unsubstantiated) for a particular product or initiative.

On the contrary, ROI analysis aims to achieve clarity in the decision-making process. For example, from a technical perspective, we generally want to increase a software product's quality because fixing existing software takes valuable time away from developing new software. But how much investment in software quality is desirable? When should we invest, and where?

A robust economic and strategic analysis can help answer these questions. If a firm's strategy to achieve competitive advantage is based on higher customer satisfaction and corresponding higher prices, then it can argue for a certain level and type of investment in quality. A competitive strategy based on lower production costs and lower

0740-7459/04/$20.00 © 2004 IEEE

prices might lead to targeting the investment in quality to specific parts of the software product or service. In this way, the value of increasing software quality is framed in terms that senior management can understand and use to make informed decisions. But even technical personnel can benefit from familiarity with this type of analysis. From the new agile development methodologies that explicitly incorporate notions of customer value to the latest asset-based repositories such as product lines, it's clear that the software industry currently views few initiatives from a purely technical perspective.

Learning the business

It's not the widespread promotion of ROI in the software industry that is misplaced, but the use of a single, narrowly defined concept as a proxy for an entire hierarchy of activities concerned with financial and strategic analysis. These activities fall essentially into four levels, each driving the next:

- *Business strategy:* Selecting markets in which to participate and formulating strategies for competing in those markets
- *Valuation:* Analyzing the economic value of projects executed in the pursuit of that business strategy
- *Cost-benefit analysis:* Translating measured or estimated data into monetary terms (for example, labor costs), forming the basis for valuation
- *Metrics:* Measuring the parameters (such as programmer time) that form the basis for cost and benefit analysis

Not surprisingly, the software engineering community has done an enormous amount of work in the latter two areas while effectively neglecting the first two. We feel comfortable with gathering and interpreting metrics, which are naturally "close to the code," whereas valuation and competitive strategy seem to belong to the unfamiliar realm of corporate finance. Unfortunately, our educational system neglects to expose software engineers to business fundamentals and doesn't instill a good understanding of the role of technical projects in the context of overall business imperatives. So, we're often not well equipped to elaborate the business case behind our technical vision.

Yet the fundamentals of financial and strategic analysis are within any software engineering professional's grasp. Few disciplines offer better preparation—software engineers can understand and apply abstract concepts, can think clearly and logically, and last but not least, have computational skills. It's ironic that the legions of programmers constructing the financial and strategic analysis tools that businesses around the world use don't apply these same tools to their own products and processes.

Analyzing and creating economic value

It's equally ironic that ROI has acquired the reputation of being a vague and slippery marketing device in the software community, because the definition of ROI used in finance (and in this special issue) is anything but—it's the ratio of net benefits to costs, or

(Benefits – Costs)/Costs.

The ROI calculation organizes a project's costs and benefits ("cash flows" in finance jargon) into a useful profitability measure. But this measure alone doesn't capture two essential ingredients of any serious economic analysis: time and risk. Without the dimension of time, we can't compare the economics of short-lived versus long-lived projects; without accounting for risk, we can't compare a project with its riskier competitor. The rigorous treatment of time and risk brings an analysis firmly into the domain of *valuation*.

At the heart of modern valuation is the only logical and universally accepted definition of economic value: *net present value*. NPV analysis levels the playing field for costs and benefits occurring near and far in the future by aligning them all to a single time frame in the present. Is early, one-time delivery preferable to a long-term subscription offering? Should I pour my resources into that one-time, quick-payback project now or invest in a more ambitious program of systematic reuse? Valuation can help make sense of the numbers, giving proper weight to costs and benefits occurring at different points in time.

By explicitly modeling the *time value of money*, NPV introduces another fundamental concept into valuation. The return generated

Further Reading and Resources

Basic financial and economic concepts underlying ROI appear in introductory corporate finance books—see, for example, relevant chapters on capital budgeting and risk and return in *Fundamentals of Corporate Finance*.[1] An *Annals of Software Engineering* article provides a concise introduction to these concepts and the fundamentals of option pricing theory in the software engineering context.[2] Michael Porter wrote the standard text on competitive strategy.[3]

Barry Boehm wrote a comprehensive reference on software engineering economics,[4] which, along with the classic book on cost estimation,[5] provides a solid foundation for understanding ROI in the software context. For an up-to-date perspective on ROI in the software industry, read Boehm's article on value-based software engineering,[6] which defines a high-level, multistakeholder, multifacet framework for capturing value in the full spectrum of software engineering activities, from requirements engineering to risk management.

Donald Reifer's book is a high-level practitioner resource for software business case analysis.[7] It takes a pragmatic approach to using numbers to justify software-intensive projects and contains many examples. As such, it complements this issue's two valuation articles.

An excellent reference on ROI for software quality is Khaled El-Emam's book,[8] which includes several ROI calculation examples for inspection, risk assessment, test-driven development, and defect detection. It also provides international benchmarks for these quality practices. Capers Jones' book is another comprehensive account of industry benchmarks for software process improvement and associated practices.[9] For a shorter account, refer to Steve McConnell's summary of ROI benchmarks for process improvement,[10] which complements the benchmarks provided in this issue's "Measuring the ROI of Software Process Improvement."

Most existing literature on software engineering economics focuses on process or business decisions, so works that address technical decisions are scarce. Representative resources in the latter category include "Software Design as an Investment Activity"[11] and *Evaluating Software Architectures*,[12] both of which provide theoretical and practical treatments that tackle ROI in design and architecture decisions.

For the research-oriented, we suggest the ICSE (International Conference on Software Engineering) roadmap document by Barry Boehm and Kevin Sullivan.[13] This reference provides an agenda of research topics pertaining to ROI in the software engineering context.

Interest in risk management has been increasing in response to the technology sector's growing volatility. From an ROI perspective, of particular interest is an approach referred to as *real options*. This approach reconciles financial and strategic viewpoints to support decision making in the face of uncertainty—in particular, for valuing flexibility. Recent applications of this approach in the software engineering context include the assessment of value propositions for agile software development,[14] valuation of staged software development projects subject to multiple sources of uncertainty,[15] and the quantification of the strategic benefits of platform investments.[16] The last reference is of particular interest because it complements the ROI approach proposed in this issue's "Calculating ROI for Software Product Lines."

Finally, the Economics-Driven Software Engineering Research workshops also constitute a valuable resource (www.edser.org). EDSER proceedings contain many short position papers on using value-based approaches in a variety of software engineering problems ranging from technical decisions to business and process decisions. We encourage interested readers to attend the EDSER workshops, which have been collocated with ICSE since 1998.

References

1. S.A. Ross et al., *Fundamentals of Corporate Finance*, Times Mirror Professional Publishing, 1996.
2. J.M. Favaro, K.R. Favaro, and P.F. Favaro, "Value Based Software Reuse Investment," *Annals of Software Eng.*, vol. 5, 1998, pp. 5–52.
3. M.E. Porter, *Competitive Strategy*, The Free Press, 1980.
4. B. Boehm, *Software Engineering Economics*, Prentice Hall, 1981.
5. B. Boehm et al., *Software Cost Estimation with Cocomo II*, Prentice Hall, 2000.
6. B. Boehm, "Value-Based Software Engineering," *Software Eng. Notes*, vol. 28, no. 2, 2003.
7. D. Reifer, *Making the Software Business Case*, Addison-Wesley, 2002.
8. K. El-Emam, *The ROI from Software Quality: An Executive Briefing*, Ottawa Software Quality Assoc., 2004.
9. C. Jones, *Software Assessments, Benchmarks, and Best Practices*, Addison-Wesley, 2002.
10. S. McConnell, "Business Case for Better Software Practices," *Professional Software Development*, Addison-Wesley, 2004, pp. 111–122.
11. K.J. Sullivan et al., "Software Design as an Investment Activity: A Real Options Perspective," *Real Options and Business Strategy: Applications to Decision Making*, L. Trigeorgis, ed., Risk Books, 1999.
12. P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architectures: Methods and Case Studies*, Addison-Wesley, 2002.
13. B. Boehm and K.J. Sullivan, "Software Engineering Economics: A Roadmap," *The Future of Software Engineering*, A. Finkelstein, ed., ACM Press, 2000.
14. H. Erdogmus and J. Favaro, "Keep Your Options Open: Extreme Programming and the Economics of Flexibility," *Extreme Programming Perspectives*, L. Williams et al., eds., Addison-Wesley, 2002, pp. 503–552.
15. H. Erdogmus, "Valuation of Learning Options in Software Development Under Private and Market Risk," *The Eng. Economist*, vol. 47, no. 3, 2002, pp. 304–353.
16. J.M. Favaro and K.R. Favaro, "Strategic Analysis of Application Framework Investments," *Building Application Frameworks: Object Oriented Foundations of Framework Design*, M. Fayad and R. Johnson, eds., John Wiley & Sons, 1999, pp. 567–597.

by money earning interest over time becomes a baseline ROI against which we can compare alternative uses for that money. This *opportunity cost of money* effectively represents a hurdle that a competing investment opportunity must overcome to be justifiable (otherwise, you might as well put your money in the bank). The opportunity cost of money and its role in creating or destroying economic value is the core principle driving the best management practices available today.

However, time isn't the only factor determining the opportunity cost of money. If a software development project promises the same ROI as money sitting in the bank, then an investor would insist on a discount for taking the additional risk—the higher the perceived risk, the higher the demanded discount. Treating risk in modern valuation goes far beyond the intuitive notions underlying typical project-level risk management programs in the software industry. Valuation explicitly quantifies the level of reward that an investor may demand for taking on risk. Furthermore, it does this while reconciling a project's specific risks with the inevitable, systematic risks that the overall economy thrusts upon it.

Valuation is an important tool for analyzing economic value, but business strategy is the key to creating it. Yet even more so than valuation, most software engineers seem to see business strategy as a riddle wrapped in mystery inside an enigma. Given the careless, overhyped, and even contradictory use of mantras such as "time to market" and "explosive growth" in recent years, the confusion is understandable.

Properly understood and executed, however, business strategy development is a methodical, disciplined exercise based on solid and rational principles. It's thus worthy of study by any software professional—especially given software's highly strategic nature in today's business environment. Analyzing market structure is an essential component of strategy development. Is the average market participant profitable? Are there high barriers to entry? Answering questions such as these can help strategists wisely select markets in which to participate and avoid jumping blindly onto the latest bandwagon. Competitive strategy development in a chosen market isn't a matter of merely "playing to win" but involves making hard choices: Can high market share be pursued profitably at these prices? Is the nature of the service such that we can differentiate our offering and justify higher prices? Will shorter time to market result in a sustainable competitive advantage or just an ephemeral flash in the pan?

The basic principles of market selection and competitive strategy, accessible to any software professional, were violated again and again during the boom years. It's unlikely that the future will be so forgiving.

In this issue

From 21 submissions, we selected six articles, each addressing one of the top three levels of ROI-related activities—cost-benefit analysis, valuation, or business strategy. We aim to provide, through illustrative examples, a broad perspective on

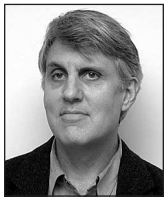
- How different kinds of software organizations define and use ROI in different sectors and for different process- and product-related decisions
- Emerging practical approaches for reasoning about ROI at the top three levels

"Calculating ROI for Software Product Lines" and "Measuring the ROI of Software Process Improvement" focus on cost-benefit analysis. The first develops a business rationale and classic ROI model for migrating a set of software products to a product line, while the latter addresses ROI specifically in the software process improvement domain. The next two articles focus on valuation. "The Incremental Funding Method: Data-Driven Software Development" describes a valuation-based method for release planning in the context of incremental development. "Value Creation and Capture: A Model of the Software Development Process" describes a visual, functional model for analyzing a software development project's value. To illustrate the broad nature of strategic issues pertaining to ROI, we selected two articles of very different flavors. "The ROI of Software Dependability: The iDAVE Model" describes a method for exploring software dependability strategies based on empirical cost and quality models. "Marketplace Issues in Software Planning and Design" gives a high-level overview of marketplace factors and strategies in commercial

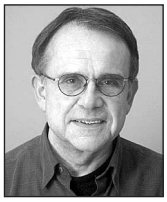
Most software engineers seem to see business strategy as a riddle wrapped in mystery inside an enigma.

About the Authors

Hakan Erdogmus is a senior research officer at the National Research Council, Canada. His research interests include software engineering economics, agile software development, and collaborative software development. He received his PhD in telecommunications from INRS, Université du Québec. Contact him at NRC Inst. for Information Technology, Montreal Rd., M-50, Ottawa K1A OR6, Canada; hakan.erdogmus@nrc-cnrc.gc.ca.



John Favaro is the founder of Consulenza Informatica in Pisa, Italy. His research interests include software reuse, agile methods, and the value-based management of information technology. He received his MS from the University of California, Berkeley. He is a founding member of the International Society for the Advancement of Software Education. Contact him at Consulenza Informatica, Via Gamerra 21, Pisa, Italy; jfavaro@tin.it.



Wolfgang Strigel is the founder and president of Software Productivity Center, a consulting and products company, and of QA Labs, a contract testing company. His interests include collaborative software development, process improvement, project estimation, testing, and software engineering economics. He has an MSc in computer science from McGill University and an MBA from Simon Fraser University. Contact him at strigel@spc.ca.

software development, such as pricing, standards, switching costs, competition, and project structuring.

Although these articles provide concrete ideas of how to reason about ROI in the context of software development, they don't span the whole ROI space. We thus identify additional resources in the related sidebar.

To the practicing software engineer, finance and business strategy might seem to be distant, irrelevant worlds. But ignoring them isn't in our best interest, because they draw closer every day and offer challenges that are as intellectually demanding and satisfying as any in our own field. We hope this special issue raises ROI awareness in the software community.