A Follow-Up Reflection on Software Process Improvement ROI

Rini van Solingen, *Delft University of Technology*

Our discipline must shift toward value-based software engineering, because we're obliged to prove our contributions to the financial bottom line.

In the May/June 2004 *IEEE Software* special issue on return on investment (ROI), I presented measurement results for the ROI of software process improvement (SPI).[1] My article made three main contributions.

First, I provided a detailed overview of publications containing real-life measurement results from practical applications of SPI, in which I measured the ROI. My study included 20 cases, with an average ROI of 7 and a median of 6.6. This indicates that SPI's net profit seems to be approximately US$7 for every dollar invested. However, I found no published cases in which SPI investments resulted in a measurable loss; furthermore, the ROI bandwidth was large (between 1.5 and 19). This indicates that the actual ROI of an SPI investment seems hard to really guarantee up front.

Second, I showed that benefits are just as easy to quantify as costs. Cost measurements are always based on an agreement about how to measure and quantify costs. Such an agreement can also serve as the basis for measuring benefits. My article contained data from two real-life projects that had made such cost and benefit measurements and calculated ROI.

Finally, I concluded that expressing "value" is crucial. Software engineering and its improvement are often major investments for organizations. Investments must be profitable. Because different people in different roles share one generic term for value—money, I recommended expressing any software engineering effort and its benefits in financial terms.

So how does the world look five years after that article's publication? Surprisingly, the number of studies containing ROI data on SPI hasn't grown much. It almost seems the article made an end to the investigation of how SPI contributes to the bottom line. However, this situation has provided a reasonable opportunity for a follow-up article. After all, the question seems more relevant than ever: Why don't we measure the financial value of the undertakings in our discipline?

Rini van Solingen's 2004 *IEEE Software* article, "Measuring the ROI of Software Process Improvement" (May/June 2004, pp. 32–38), attracted much reader interest because it provided hard data on the bottom-line implications of various software process initiatives. The article made our 25th-Anniversary Top Picks list back in January (pp. 9–11). It was among the handful of the selected articles for which we thought an update would be timely and welcome. Rini obliged by providing us reflections on this earlier work and some more data, courtesy of a colleague of his.

—*Hakan Erdogmus*

Measuring Value

It seems obvious that measuring an investment's

0740-7459/09/$26.00 © 2009 IEEE

Management Commitment

For an illustration of our failure to express value, look at the "management commitment" success factor. Almost every publication on software-process-improvement success factors indicates that management commitment is crucial.[1] It's commonly accepted that we must establish management commitment before starting an improvement initiative. But is this really true? Or are we attempting to cover up one of our own basic mistakes? I think the latter is the case, for two reasons.

First, if there's one way to get management committed, it's by creating profit. If expenditures are clearly worth the money, you'll get management commitment with it for free. As long as activities guarantee large profits, why would management stop them? Our emphasis on establishing management commitment seems to result from our own failure to prove financial value in the first place.

Second, if management supports an investment, we still have the responsibility to show whether that investment is actually worth the money. Unfortunately, instead of actually measuring financial benefits and contributions to the bottom line, we advocate management commitment. We seem to think we can delegate the responsibility for showing value to our management.

Reference

1. M. Niazi, D. Wilson, and D. Zowghi, "Critical Success Factors for Software Process Improvement Implementation: An Empirical Study," *Software Process: Improvement and Practice*, vol. 11, no. 2, 2006, pp. 193–211.

value should be a logical step. However, this is hardly done in our discipline (see the sidebar). One main reason seems to be historical; originally, IT was a relatively small investment, often part of a much larger investment. However, with increased dependency on IT and with IT often becoming the primary process in, for example, financial institutions or the core component in, for example, electronic products, we need to reconsider this position. This lack of value measurement seems incorporated in not only our practices but also our research. Ramesh Venkataraman and his colleagues, for example, showed that fewer than 0.5 percent of papers in the leading computer science research journals employed methods using industrial measurements (case studies, data analysis, and field studies).[2]

Barry Boehm has argued for shifting focus to *value-based software engineering*.[3] This approach integrates value considerations into the full range of existing and emerging software engineering principles and practices. We need to incorporate a measurement culture into software engineering that focuses on the value we deliver—at the beginning of, during, and after our undertakings.

It's feasible to include value measurements and data in our discipline. For example, David Rico's research on *business value* provides detailed insights on economic indicators for people considering a specific software development or process improvement method (see Table 1).[4] Some people might dispute such quantitative models and doubt their external validity. However, these models and numbers have (at least) a strong indicative value. They help practitioners compare approaches by making clear what financial benefits they can reasonably expect.

Quantify Intangible Benefits Too

But why focus on the money side alone? Isn't it true that most benefits of SPI aren't financial and that it contributes much more through intangible benefits? Yes, that seems valid in a way. Many benefits of SPI contribute only indirectly to an organization's bottom line. Kevin Hyde and David Wilson presented an overview of SPI's intangible benefits, such as less stress, limited overtime, improved management communication, ability to change, fewer resignations, and improved morale.[6]

Unfortunately, Hyde and Wilson didn't express these benefits in financial values. They seem to have followed the common, mistaken assumption that you can't express intangible benefits in financial terms. The contrary is true: you can just as easily measure benefits in financial terms as you can costs, as long as you define an agree-

Table 1

Return on interest (ROI) metric examples* of software process improvement methods for a four-person team implementing 10,000 LOC[4]

Method	Cost ($)	Benefit ($)	Benefit/cost	ROI (%)	Net present value ($)	Break-even point ($)
Inspections	82,073	2,767,464	34:1	3,272	2,314,261	51,677
Personal Software Process	105,600	4,469,997	42:1	4,133	3,764,950	945
Team Software Process	148,400	4,341,496	29:1	2,826	3,610,882	5,760
SW-CMM	311,433	3,023,064	10:1	871	2,306,224	153,182
ISO 9001	173,000	569,841	3:1	229	320,423	1,196,206
CMMI	1,108,233	3,023,064	3:1	173	1,509,424	545,099

*For the most recent update of the ROI metrics, including agile methods, see The Business Value of Agile Software Methods.[5]

ment on the measurement procedure.[1] For example, if a manager expresses that increasing morale is important, what is he or she willing to pay for it? If this manager is responsible for 1,000 people and will pay $1,000 per employee for a 50 percent increase in morale, achieving this increase will reap a benefit of least 1 million dollars (you can measure benefits in terms of a purchase value). Of course, such numbers might not be extremely accurate; however, they're absolutely useful to quantify the ROI of an SPI investment made in that organization.

Consider what will happen when software developers accept that they can measure intangible benefits just as easily as costs. It will enable the quantification and economic validation of any investment done in our discipline. We'll no longer have to convince others on the basis of our faith in expertise, because we'll have access to the numbers that show where we create value. This has already happened in other disciplines—for example, in the manufacturing discipline (where profit margins seem to shrink year by year). So, manufacturing needed to integrate economic quantification throughout the discipline, and its improvement methods had to incorporate such quantification as well. Why not adopt such value-based improvement methods in software engineering?

Six Sigma as a Strategy for Improving Software Engineering

The most well-known improvement approach for manufacturing is probably Six Sigma. This approach deploys a set of quality management methods (strongly depending on statistical techniques) and creates a special organizational infrastructure of people who are experts in applying these methods. It triggers a never-ending series of improvement projects that stabilize process performance by revealing causes of problems and resolving them. All these projects aim to produce quantified financial benefits, in a fixed sequence of steps at a fixed pace. Because of this strong focus on producing financial value, every aspect of Six Sigma contains estimation and measurement of value. Projects start with a charter summarizing the problem to solve and the value to create, and end with deployment actions to ensure the continued delivery of benefits and to measure those benefits.

Embracing Six Sigma in software engineering will

- incorporate value measurement directly into everything we try to improve,
- provide knowledge on how our activities actually contribute to the bottom line,
- lead to management commitment because we'll be able to indicate and measure value,
- challenge technologies and methods to prove real-life benefits before they're deployed,
- lead to repeated successes because successful improvements in one area will be put on the top of the list for repeating in others,
- help stabilize process performance by shifting the focus from increasing an indicator's mean to decreasing its spread, and
- establish statistical and quality management skills for the people leading software development and process improvement.

Although adoption of Six Sigma in our discipline is still relatively low, attention to it is increasing.[7,8] Adoption of Six Sigma principles in software engineering and process improvement will cause a breakthrough because it will emphasize the responsibility to measure value.

Because software engineering is so strongly related to the success and failure of business, we must find a way to measure value. Without such measurements, putting a value to an investment is left to people's intuition and not to the proven facts. The adoption of Six Sigma might just be the catalyst we need in our discipline. Software companies that embrace Six Sigma to structure their improvements will outclass competitors because of their skill in having their improvement efforts contribute directly to the financial bottom line.

In stating that, I'm pretty much making the same mistake I criticized in this article, because I actually can't prove this value by extensive measurements for software engineering. However, much evidence from other disciplines indicates that Six Sigma is at least a promising opportunity for ours.

One thing, however, is clear: our discipline's continuing growth urges us to shift toward value-based software engineering. The numbers we find will then speak for themselves.

About the Author

Rini van Solingen is an independent consultant and a part-time associate professor in globally distributed software engineering in the Delft University of Technology's Informatics Department. Van Solingen has an MSc in computer science from that university and a PhD in management science from the Eindhoven University of Technology. Contact him at d.m.vansolingen@tudelft.nl; he'll be pleased to hear from you.

References

1. R. van Solingen, "Measuring the ROI of Software Process Improvement," *IEEE Software*, May/June 2004, pp. 32–38.
2. V. Ramesh, R.L. Glass, and I. Vessey, "Research in Computer Science: An Empirical Study," *J. Systems and Software*, vol. 70, nos. 1–2, 2004, pp. 165–176.
3. B. Boehm, "Value-Based Software Engineering: Overview and Agenda," *Value-Based Software Engineering*, S. Biffl et al., eds., Springer, 2005, pp. 3–14.
4. R. van Solingen and D.F. Rico, "Calculating Software Process Improvement's Return on Investment," *Advances in Computers 66: Quality Software Development*, Elsevier, 2006, pp. 2–43.
5. D.F. Rico, H.H. Sayani, and S. Sone, *The Business Value of Agile Software Methods: Maximizing ROI with Just-in-Time Processes and Documentation*, J. Ross, 2009.
6. K. Hyde and D. Wilson, "Intangible Benefits of CMM-Based Software Process Improvement," *Software Process: Improvement and Practice*, vol. 9, no. 4, 2004, pp. 217–228.
7. J. Siviy, M.L. Penn, and E. Harper, "Relationships between CMMI and Six Sigma," tech. note CMU/SEI-2005-TN-005, Software Eng. Inst., Carnegie Mellon Univ., 2005.
8. C.B. Tayntor, *Six Sigma Software Development*, CRC Press, 2007.