The return on investment in improved software practices is well documented. In 1994, James Herbsleb reported that the average “business value” (roughly the same as ROI) for 13 organizations that took on CMM-SW-based improvement programs was about 5 to 1, with the best organizations realizing returns of 9 to 1.\textsuperscript{1} In 1995, Neil C. Olsen reported similar returns for organizations that made significant investments in staffing, training, and work environments.\textsuperscript{2} In 2000, Capers Jones reported that the ROI from process improvement could easily go into double digits (meaning returns greater than 10 to 1).\textsuperscript{3} A recent analysis by Watts Humphrey found that the ROI for improved software practices could be in the neighborhood of 4 to 1.\textsuperscript{4}

**Indirect benefits are even more significant**

The ROI figures in the published literature are based on operational savings—that is, on reducing development cost per line of code written or per function point delivered. Although these savings are impressive, the greater business benefit might arise from the significant *indirect* returns that arise from improved software practices. Better software practices improve predictability of costs and schedules, reduce risk of cost and schedule overruns, provide early warning of problems, and support better management.

Many organizations that have focused on improving their software practices have reported improvements in predictability similar to the results in Figure 1.\textsuperscript{5} For a software products company, what is the business value of improving schedule estimation accuracy from plus or minus 100 percent to plus or minus 10 percent? What is the value of being able to make a commitment to customers six to 12 months in advance of a scheduled completion date, with high confidence of delivering on that commitment?

For a company that develops custom software, what is the business value of being able to provide a fixed price bid with high confidence that the project will not significantly overrun the bid?

For a retail sales organization, what is the value of being able to plan cutover to a new system with pinpoint accuracy? What is the value of knowing with confidence that cutover will occur 1 October, as planned, with little risk of overrunning to 15 November or 1 December?

Unlike the operational benefits that most of the industry literature has focused on, these indirect benefits open the door to additional revenue opportunities. These benefits are based not on reducing costs, but on increasing access to additional business. For top decision makers in organizations, these indirect benefits are often more compelling than the direct, operational benefits.

**Organizational challenge**

Considering the strong—even compelling—case for improving software practices, it might seem surprising that some organizations have not made a commitment to use best practices. I have recently been think-
ing a lot about why best practices aren’t used. Several factors seem to be in play.

First, there is a basic technology transfer issue. Many software development best practices have been available for decades, but only a few companies use them, and undergraduate programs have not generally taught these best practices. The scarcity of experienced users of these practices limits the rate at which current users can train new users. Although a person might reasonably assume that the average software organization’s capability is halfway between the worst organization’s capability and the best’s, in reality, the average software organization’s practices are much closer to the worst organization’s practices than the best’s. The result is that software developers who work in average organizations—which includes most developers—have never seen a really well-run software project, much less a really well-run software organization. The software industry faces the problem of bootstrapping best practices into common usage because of the transfer issue. Many software development best practices have been in play.

A final factor is that many organizations push responsibility for software development improvement down to the project level. In reviewing the “effort multiplier” factors in the Cocomo II estimation model,7 I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors in Cocomo II that would have shifted too much focus away from generating revenue. For a time, improved software practices seemed to be more of a distraction than a help.

A final factor is that many organizations push responsibility for software development improvement down to the project level. In reviewing the “effort multiplier” factors in the Cocomo II estimation model,7 I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors in Cocomo II that would have shifted too much focus away from generating revenue. For a time, improved software practices seemed to be more of a distraction than a help.

A final factor is that many organizations push responsibility for software development improvement down to the project level. In reviewing the “effort multiplier” factors in the Cocomo II estimation model,7 I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors in Cocomo II that would have shifted too much focus away from generating revenue. For a time, improved software practices seemed to be more of a distraction than a help.

A final factor is that many organizations push responsibility for software development improvement down to the project level. In reviewing the “effort multiplier” factors in the Cocomo II estimation model,7 I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors in Cocomo II that would have shifted too much focus away from generating revenue. For a time, improved software practices seemed to be more of a distraction than a help.

A final factor is that many organizations push responsibility for software development improvement down to the project level. In reviewing the “effort multiplier” factors in the Cocomo II estimation model,7 I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors in Cocomo II that would have shifted too much focus away from generating revenue. For a time, improved software practices seemed to be more of a distraction than a help.

A final factor is that many organizations push responsibility for software development improvement down to the project level. In reviewing the “effort multiplier” factors in the Cocomo II estimation model,7 I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors in Cocomo II that would have shifted too much focus away from generating revenue. For a time, improved software practices seemed to be more of a distraction than a help.

A final factor is that many organizations push responsibility for software development improvement down to the project level. In reviewing the “effort multiplier” factors in the Cocomo II estimation model,7 I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors in Cocomo II that would have shifted too much focus away from generating revenue. For a time, improved software practices seemed to be more of a distraction than a help.

A final factor is that many organizations push responsibility for software development improvement down to the project level. In reviewing the “effort multiplier” factors in the Cocomo II estimation model,7 I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors in Cocomo II that would have shifted too much focus away from generating revenue. For a time, improved software practices seemed to be more of a distraction than a help.

A final factor is that many organizations push responsibility for software development improvement down to the project level. In reviewing the “effort multiplier” factors in the Cocomo II estimation model,7 I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors in Cocomo II that would have shifted too much focus away from generating revenue. For a time, improved software practices seemed to be more of a distraction than a help.

A final factor is that many organizations push responsibility for software development improvement down to the project level. In reviewing the “effort multiplier” factors in the Cocomo II estimation model,7 I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors in Cocomo II that would have shifted too much focus away from generating revenue. For a time, improved software practices seemed to be more of a distraction than a help.

A final factor is that many organizations push responsibility for software development improvement down to the project level. In reviewing the “effort multiplier” factors in the Cocomo II estimation model,7 I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors in Cocomo II that would have shifted too much focus away from generating revenue. For a time, improved software practices seemed to be more of a distraction than a help.

A final factor is that many organizations push responsibility for software development improvement down to the project level. In reviewing the “effort multiplier” factors in the Cocomo II estimation model,7 I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors in Cocomo II that would have shifted too much focus away from generating revenue. For a time, improved software practices seemed to be more of a distraction than a help.

A final factor is that many organizations push responsibility for software development improvement down to the project level. In reviewing the “effort multiplier” factors in the Cocomo II estimation model,7 I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors in Cocomo II that would have shifted too much focus away from generating revenue. For a time, improved software practices seemed to be more of a distraction than a help.

A final factor is that many organizations push responsibility for software development improvement down to the project level. In reviewing the “effort multiplier” factors in the Cocomo II estimation model,7 I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors in Cocomo II that would have shifted too much focus away from generating revenue. For a time, improved software practices seemed to be more of a distraction than a help.

A final factor is that many organizations push responsibility for software development improvement down to the project level. In reviewing the “effort multiplier” factors in the Cocomo II estimation model,7 I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors in Cocomo II that would have shifted too much focus away from generating revenue. For a time, improved software practices seemed to be more of a distraction than a help.

A final factor is that many organizations push responsibility for software development improvement down to the project level. In reviewing the “effort multiplier” factors in the Cocomo II estimation model,7 I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors in Cocomo II that would have shifted too much focus away from generating revenue. For a time, improved software practices seemed to be more of a distraction than a help.

A final factor is that many organizations push responsibility for software development improvement down to the project level. In reviewing the “effort multiplier” factors in the Cocomo II estimation model,7 I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors in Cocomo II that would have shifted too much focus away from generating revenue. For a time, improved software practices seemed to be more of a distraction than a help.

A final factor is that many organizations push responsibility for software development improvement down to the project level. In reviewing the “effort multiplier” factors in the Cocomo II estimation model,7 I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors in Cocomo II that would have shifted too much focus away from generating revenue. For a time, improved software practices seemed to be more of a distraction than a help.

A final factor is that many organizations push responsibility for software development improvement down to the project level. In reviewing the “effort multiplier” factors in the Cocomo II estimation model,7 I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors in Cocomo II that would have shifted too much focus away from generating revenue. For a time, improved software practices seemed to be more of a distraction than a help.
development, personnel continuity, process maturity, and so on—can be influenced by the organization but not by individual projects.

What can you do?

We could hope that upper management, sales, and marketing staff would read every issue of IEEE Software to cover or educate themselves about the finer nuances of software development some other way. But this isn’t likely to happen, so leading software practitioners have an ongoing responsibility: the education of nontechnical software project stakeholders. Software practitioners sometimes perceive upper management and other nontechnical staff to be blocking the use of better practices. We complain that they fail to support better practices or even undermine them. I’ve generally found, however, that upper management, sales, marketing, product support, and other personnel are receptive to improved software practices when I take the time to explain those practices to them. Indeed, they are acutely aware of the problems caused by current practices and are eager to hear how they can help improve software projects.

What have you done to educate executives about better software practices? What has worked well for you? I’d love to hear your comments at stevemcc@construx.com.

References


Call for Articles: Industry Experience Reports

Experience reports with lessons learned in industry are one way to share successes or failures with others who likely face similar situations. We invite you to report on a technology or process you introduced in your company, analyze its impact, and explore what you would do differently the next time.

Articles should be 2,000 to 2,400 words with each illustration counting as 200 words. We also encourage you to submit up to 10 short bullet points on lessons learned and references to related Web sites. Submissions are reviewed by members of our Industry Advisory Board and are subject to editing for style, clarity, and space.

For detailed author guidelines, see computer.org/software/genres.htm#IndustryExperienceReport or contact software@computer.org. Submissions are accepted at any time.

Associate Editor: Wolfgang B. Strigel, Software Productivity Center, strigel@spc.ca.