Editor in Chief ■ Warren Harrison ■ Portland State Univ. ■ warren.harrison@computer.org

What Do Software Developers Need to Know about Business?

Warren Harrison

A recent discussion I had with colleagues from my university's business school and computer science department focused on identifying the most critical knowledge for software developers. My computer science colleagues' perspective was quite interesting. They acknowledged that once a software developer has managed to climb into a second- or third-level management position, maybe an MBA wouldn't be such a bad idea. But for the most part, they held the strong belief that anyone smart enough to be a computer science graduate must be able to easily pick up this "business stuff" on the side.

Type a or type b?

I felt the same quick surge of irritation that I usually get when dealing with students who want to bypass prerequisite courses and enroll directly into an upper-division software engineering course. They seem to think that software engineering can't be all that difficult compared to the mysteries of homomorphisms and context-free grammars. It's either (a) the height of conceit or (b) the height of ignorance about the depth of the field to suggest that, without serious study, someone could pick up a field that has taken others years to master. Over the years, I've come to realize that (b) is more common than (a).

So, whenever I hear someone talk about either business or computing issues as things to be "picked up on the job," I can't help but wonder whether the person is "type a" or "type b." That's not to say that someone can't learn material from either discipline through self-study—but it requires a serious application of discipline and effort.

Understanding the context

I'm less concerned about those who have chosen to pursue a management career path. Over time, I've come to believe there's a real need for the average software developer to understand and appreciate the economic context in which their company operates.

Without a thorough understanding of the business context, we tend to operate in a fog, believing things happen arbitrarily by chance and can't be predicted. Primitive cultures' lack of understanding of physical phenomena led to a belief that people became ill, soldiers won or lost battles, and storms devastated villages because of angry "gods." Individuals felt a loss of control over their daily lives and resorted to illogical attempts to placate these "gods" through sacrifice and the construction of grand temples.

I've observed developers' bitter resentment when they try to come to grips with senior management's seemingly illogical decisions and actions. Often, as in ancient mythology, they attribute these actions to angry "gods" who arbitrarily cancel projects, cut staff, and enforce technical mandates.

The fact is, modern, multinational businesses operate within a set of well-understood phenomena. If you don't understand discounted cash flow, hurdle rate, or amortization of intangibles, many management decisions inflicted on you might appear to be mere rolls of the dice or the wrath of angry "gods." Here are a few principles that can help put management decisions into a clearer context.

An investment is simply an investment

Many professions require a certain amount of detachment from events. For example, scientists are renowned for their ability to unemotionally view and record phenomena that would have lay people running for cover. The journalist's credo is to never get personally involved with the story. Police officers quickly learn to detach themselves emotionally from grisly murders and deadly traffic accidents.

If I had to put my finger on the single thing of most value that I received from my years studying for an accounting degree, it would be the ability to view investments as just that—investments. Nowhere is this better exemplified than in the concept of sunk cost that's mercilessly hammered into cost-accounting students. This might be the hardest concept for laypeople to understand. "But we've already invested so much money into this project" is a common lament among software developers when they hear that their project has been cut.

From the savvy business person's perspective, costs that have been expended in the past have no role in planning for costs in the future.

Tomorrow's dollar is worth (far) less than today's

Most of us are vaguely aware that getting money now is better than having to wait for it, but we seldom go to the trouble of quantifying exactly how much better. Business runs on the concept of quantified present value—the value of a sum received in the future, measured in "today's dollars." We can even factor in risk premiums to reflect the investment's risks and opportunity costs.

Most business people I know think in terms of risk-adjusted present value. As a result, long-term initiatives with even moderately uncertain payoffs quickly lose their attractiveness. This is especially true when other investment opportunities exist.

Capital budgeting

I've endured more than one meeting in which technical people propose projects with seemingly no clue that the money spent developing a software product could just as easily be invested in a bank, gold bullion, or opening a Kentucky Fried Chicken franchise. You can usually identify these naive champions right away: they're the ones with a laundry list of poorly quantified, intangible "benefits" that will accrue if only management would open that squeaky purse and toss out a few million.

Of course, these unfortunate souls usually try to dress up their presentations with "business speak." I once heard someone speak glowingly of the ROI expected from a project he was proposing. It quickly became clear that he didn't understand what ROI stood for when his slides started displaying ROI figures in terms of months. It turned out that he had simply plugged some numbers into a textbook equation without understanding what he was doing. Subsequently, he and his team were aghast when the tight-fisted spawn of Ebenezer Scrooge turned down the proposal.

If you don't know what your company's hurdle rate is (or worse yet, if you don't know what a hurdle rate is), do your homework before you make your pitch to the suits. At the very least, you'll impress the decision makers. At the very worst, you'll avoid making a fool of yourself proposing a project that will take two years to yield the same returns you can get at the local Savings and Loan.

Oh, and don't forget, ROI stands for return on investment, which is not the same thing as IRR (internal rate of return) or payback period.

IEEE Software

DEPARTMENT EDITORS

Bookshelf: Warren Keuffel, wkeuffel@computer.org

Design: Martin Fowler, fowler@acm.org

Loyal Opposition: Robert Glass, rglass@indiana.edu

Open Source: Christof Ebert, christof.ebert@alcatel.com

Quality Time: Nancy Eickelmann, nancy.eickelmann@motorola.com, and Jane Hayes, hayes@cs.uky.edu

Requirements: Neil Maiden, N.A.M.Maiden@city.ac.uk

Tools of the Trade: Diomidis Spinellis, dds@aueb.gr

STAFF

Senior Lead Editor
Dale C. Strok
dstrok@computer.org

Group Managing Editor
Crystal Shif

Senior Editors
Shani Murray, Dennis Taylor, Linda World

Staff Editor
Rita Scanlan

Editorial Assistants
Brooke Miner and Molly Mraz

Magazine Assistant
Hilda Hosillos, software@computer.org

Art Director
Toni Van Buskirk

Technical Illustrator
Alex Torres

Production Artist
Carmen Flores-Garvey

Executive Director
David Hennage

Publisher
Angela Burgess
aburgess@computer.org

Associate Publisher
Dick Price

Membership/Circulation Marketing Manager
Georgann Carter

Business Development Manager
Sandra Brown

Senior Production Coordinator
Marian Anderson

CONTRIBUTING EDITORS

Robert Glass, Keri Schreiner

Editorial: All submissions are subject to editing for clarity, style, and space. Unless otherwise stated, bylined articles and departments, as well as product and service descriptions, reflect the author's or firm's opinion. Inclusion in *IEEE Software* does not necessarily constitute endorsement by the IEEE or the IEEE Computer Society.

To Submit: Access the IEEE Computer Society's Web-based system, Manuscript Central, at http://cs-ieee.manuscriptcentral.com/index.html. Be sure to select the right manuscript type when submitting. Articles must be original and not exceed 5,400 words including figures and tables, which count for 200 words each.

So what can we do about it?

Many years ago, when I was in graduate school (1978–1984), it seemed that one person could know a little bit about almost every aspect of computer science. The body of knowledge was simply that small. Nowadays, with more and more subfields sprouting up, it's hard for students to know even what all the areas are, much less a little bit about each one. So, it's difficult to justify a large number of nontechnical courses simply to "gain context." Luckily, software developers can get by with far less—after all, the idea is to establish a context, not to turn out investment analysts.

Several years ago, Bruce Schafer (founder of PC-Kwik, for you old-timers) and I put together a course for the Oregon Master of Software Engineering program called "Understanding the Software Business." This class undertook the Herculean task of introducing students to the marketing, financial, and legal aspects of the software industry in 10 weeks. It involved general pricing strategies, basic macroeconomic concepts, capital budgeting issues, and intellectual property.

We found that students (all professional software developers with several years of experience) came into the class with a lack of exposure to economic and financial topics, a misunderstanding of the concepts of time value of money, and little knowledge of pricing strategies.

Although such a class clearly can't turn students into experts in any of these areas, it can provide them with the tools to better understand the business aspects of developing software and how decisions are made at the executive levels of their organizations. Such a course can go a long way toward preparing students for a career in software development and is likely to be far more valuable than many of the traditional undergraduate requirements such as differential equations, linear algebra, or physics.

Feedback welcome

What do you think? Have you had any experiences dealing with executive-level management at work? Do you find that your company uses formal capital-budgeting techniques in evaluating projects? If you have training in business, economics, or finance, have you found that it has benefited you in your role as a developer? Please write me at warren.harrison@computer.org.