Contact Editor: Dennis Taylor ■ dtaylor@computer.org

Software Engineering:

An Idea Whose Time Has Come and Gone?

Tom DeMarco

We're now just past the 40th anniversary of the NATO Conference on Software Engineering in Garmisch, Germany, where the discipline of software engineering was first proposed. Because some of my early work became part of that new discipline, this seems like an appropriate moment for reassessment.

My early metrics book, *Controlling Software Projects: Management, Measurement, and Estimation* (Prentice Hall/Yourdon Press, 1982), played a role in the way many budding software engineers quantified work and planned their projects. In my reflective mood, I'm wondering, was its advice correct at the time, is it still relevant, and do I still believe that metrics are a must for any successful software development effort? My answers are no, no, and no.

The book for me is a curious combination of generally true things written on every page but combined into an overall message that's wrong. It's as though the book's young author had never met a metric he didn't like. The book's deep message seems to be, metrics are good, more would be better, and most would be best. Today we all understand that software metrics cost money and time and must be used with careful moderation. In addition, software development is inherently different from a natural science such as physics, and its metrics are accordingly much less precise in capturing the things they set out to describe. They must be taken with a grain of salt, rather than trusted without reservation.

We welcome your letters. Send them to software@computer.org. Include your full name, title, affiliation, and email address. Letters are edited for clarity and space.

Compelled to Control

The book's most quoted line is its first sentence: "You can't control what you can't measure." This line contains a real truth, but I've become increasingly uncomfortable with my use of it. Implicit in the quote (and indeed in the book's title) is that control is an important aspect, maybe the most important, of any software project. But it isn't. Many projects have proceeded without much control but managed to produce wonderful products such as GoogleEarth or Wikipedia.

To understand control's real role, you need to distinguish between two drastically different kinds of projects:

- Project A will eventually cost about a million dollars and produce value of around $1.1 million.
- Project B will eventually cost about a million dollars and produce value of more than $50 million.

What's immediately apparent is that control is really important for Project A but almost not at all important for Project B. This leads us to the odd conclusion that strict control is something that matters a lot on relatively useless projects and much less on useful projects. It suggests that the more you focus on control, the more likely you're working on a project that's striving to deliver something of relatively minor value.

To my mind, the question that's much more important than how to control a software project is, why on earth are we doing so many projects that deliver such marginal value?

Continued on p. 95

 0740-7459/09/$25.00 © 2009 IEEE

Continued from p. 96

Can I really be saying that it's OK to run projects without control or with relatively little control? Almost. I'm suggesting that first we need to select projects where precise control won't matter so much. Then we need to reduce our expectations for exactly how much we're going to be able to control them, no matter how assiduously we apply ourselves to control.

An Unsettling Analogy

Imagine you're trying to control a teenager's upbringing. The very idea of controlling your child ought to make you at least a little bit queasy. Yet the stakes for control couldn't be higher. If you fail in your task, fail utterly, lives can be ruined. So, it's absolutely essential that you not lose your grip entirely. You're like a fencer who's learning to hold his sword as though it were a bird: too tight and the bird will be injured; too loose and it will fly away.

Now apply "You can't control what you can't measure" to the teenager. Most things that really matter—honor, dignity, discipline, personality, grace under pressure, values, ethics, resourcefulness, loyalty, humor, kindness—aren't measurable. You must steer your child as best you can without much metric feedback. It's hard, but then parenting is hard. You get a little bit of measurement in the form of school grades, and you're grateful for it. But you also know that your child's math grade is a better indicator of achievement than his Spanish grade, because math understanding is easier to measure. And his "grade" in comportment is much more likely to tell you something about the teacher than about the child.

So, how do you manage a project without controlling it? Well, you manage the people and control the time and money. You say to your team leads, for example, "I have a finish date in mind, and I'm not even going to share it with you. When I come in one day and tell you the project will end in one week, you have to be ready to package up and deliver what you've got as the final product. Your job is to go about the project incrementally, adding pieces to the whole in the order of their relative value, and doing integration and documentation and acceptance testing incrementally as you go."

This might sound like an agile-methods prescription, but I'm too far away today from the actual building of software to recommend at the methods level. Rather, I'm advocating a management approach, one that might well steer the team toward agile methods, at least toward the incremental aspects of the agile school.

So far, I've mostly discussed software engineering's metric component. How about the rest? I'm gradually coming to the conclusion that software engineering is an idea whose time has come and gone. I still believe it makes excellent sense to engineer software. But that isn't exactly what software engineering has come to mean. The term encompasses a specific set of disciplines including defined process, inspections and walkthroughs, requirements engineering, traceability matrices, metrics, precise quality control, rigorous planning and tracking, and coding and documentation standards. All these strive for consistency of practice and predictability.

Consistency and predictability are still desirable, but they haven't ever been the most important things. For the past 40 years, for example, we've tortured ourselves over our inability to finish a software project on time and on budget. But as I hinted earlier, this never should have been the supreme goal. The more important goal is transformation, creating software that changes the world or that transforms a company or how it does business. We've been rather successful at transformation, often while operating outside our control envelope. Software development is and always will be somewhat experimental. The actual software construction isn't necessarily experimental, but its conception is. And this is where our focus ought to be. It's where our focus always ought to have been.

Tom DeMarco is a principal of the Atlantic Systems Guild. Contact him at tdemarco@systemsguild.com.

IEEE Software (ISSN 0740-7459) is published bimonthly by the IEEE Computer Society. IEEE headquarters: Three Park Ave., 17th Floor, New York, NY 10016-5997. IEEE Computer Society Publications Office: 10662 Los Vaqueros Cir., PO Box 3014, Los Alamitos, CA 90720-1314; +1 714 821 8380; fax +1 714 821 4010. IEEE Computer Society headquarters: 2001 L St., Ste. 700, Washington, DC 20036. Subscription rates: IEEE Computer Society members get the lowest rate of US$51 per year, which includes printed issues plus online access to all issues published since 1988. Go to www.computer.org/subscribe to order and for more information on other subscription prices. Back issues: $20 for members, $163 for nonmembers (plus shipping and handling).

Postmaster: Send undelivered copies and address changes to *IEEE Software*, Membership Processing Dept., IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854-4141. Periodicals Postage Paid at New York, NY, and at additional mailing offices. Canadian GST #125634188. Canada Post Publications Mail Agreement Number 40013885. Return undeliverable Canadian addresses to PO Box 122, Niagara Falls, ON L2E 6S8, Canada. Printed in the USA.

Reuse Rights and Reprint Permissions: Educational or personal use of this material is permitted without fee, provided such use: 1) is not made for profit; 2) includes this notice and a full citation to the original work on the first page of the copy; and 3) does not imply IEEE endorsement of any third-party products or services. Authors and their companies are permitted to post their IEEE-copyrighted material on their own Web servers without permission, provided that the IEEE copyright notice and a full citation to the original work appear on the first screen of the posted copy.

Permission to reprint/republish this material for commercial, advertising, or promotional purposes or for creating new collective works for resale or redistribution must be obtained from IEEE by writing to the IEEE Intellectual Property Rights Office, 445 Hoes Lane, Piscataway, NJ 08854-4141 or pubs-permissions@ieee.org. Copyright © 2009 IEEE. All rights reserved.

Abstracting and Library Use: Abstracting is permitted with credit to the source. Libraries are permitted to photocopy for private use of patrons, provided the per-copy fee indicated in the code at the bottom of the first page is paid through the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923.