



Bells, Whistles, Power, and the Requirements Process

Tom DeMarco



MANY OF THE changes we've seen over the software industry's lifetime would have been easy enough to predict: systems became faster, more powerful, more ubiquitous, and more and more often transcended the boundaries of any single part of an organization, while stakeholders and end users became increasingly savvy. Who didn't see all of that coming?

On the other hand, there were some legitimate surprises. One very astute prognosticator, Robert Heinlein, wrote a book in the 1950s that described a computer so large that it occupied more than half of the interior of a giant spaceship (*Starman Jones*, Scribner, 1953). After rockets, airplanes, cars, buildings, and weapon systems all grew in size as their technology matured, why not computer hardware as well? Instead, as its power increased, the size of computer hardware decreased almost proportionately. (Even the best of the futurists can't be expected to get it all right.)

Three real surprises affecting our industry have touched on the part of the development process I've most closely tracked the matter of discovering and capturing system requirements. I've chosen here to combine a short essay on those three surprises with a review of a book that, to my mind, deals most adroitly with them. The book is *Mastering the Requirements Process: Getting Requirements Right*, by James and Suzanne Robertson (Pearson, 2013), a work that I have come to depend on as the clearest statement of how modern requirements work needs to proceed.

The First Surprise: Vanishing Technology

As we add new technological tools to our development process, our work becomes less, not more, technological in its focus.

This apparent paradox stems from the fact that any productivity improvement you make to any process reduces the total amount of work but makes the remaining work more concentrated in the areas that resist easy improvement. In software development, the parts that were susceptible to improvement through automation and tool use were technological tasks such as coding, debugging, database, and configuration control. The great savings in time spent on such tasks meant that the proportion of effort required for human communication, conflict resolution, motivation, hiring, team formation, and so forth increased. In terms first introduced by Fred Brooks, the proportion of our time spent on essence as opposed to accident increased ("No Silver Bullet—Essence and Accidents of Software Engineering," *Computer*, vol. 20, no. 4, 1987, pp. 10–19). A side effect was that the work became harder, not easier, since human interaction problems are always thornier than the kinds of activities we can hope to automate. This was a particular surprise:

continued on p. 102

continued from p. 104

adding labor-saving innovation to our process made the work more difficult, not less.

When I began my career, technologists (programmers) performed a requirements analysis prior to undertaking the real work of the project: coding and debugging. Today, requirements analysis is performed by a mixed team of systems and business people, and this is the real work of the project.

The Robertsons' book—I'll use the initials *MRP* to represent it from here forward—is particularly strong in deal-

ing with the nontechnological part of the process. For example, its approach is characterized by the following:

- *Informal methods.* Human communication needs most of all to be human, unconstrained by the kinds of formalisms methodologists and process folks find so appealing. *MRP* is only gently prescriptive, relying mostly on guidelines, tips, tricks, checklists, and heuristics.
- *Low whiz-bang.* Where a past generation of requirements texts extolled the virtues of modeling tools and computer-aided approaches, *MRP* offers up snow cards, simple paper templates, and models built of Post-It notes.

In addition, because new software has as its purpose to improve work

The Second Surprise: Power Shift

Every time a system is installed, somebody gains and somebody loses power.

I suppose this really shouldn't have been a surprise: earlier (pre-digital) system installations also led to drastic power shifts. Consider the installation of a new governmental system in the US in 1787–1788. The power winners

under the new constitution were Alexander Hamilton, James Madison, and George Washington, while the losers were antifederalists like Patrick Henry, George Mason, and Samuel Adams, and the members of the Congress of the Confederation.

But power shifts accompanying new systems in my own backyard took me by surprise. Like many of my colleagues, I felt that “enemies” of any new system were acting unprofessionally, putting their own narrow interests ahead of the organization's interests.

Today, I know that such people are not enemies at all, just potential power losers. What makes system building so complex is that the cooperation and participation of power losers (*MRP* calls them “negative stakeholders”) is absolutely essential. The book counsels

us to approach this delicate matter as an exercise in

- *Inclusiveness.* The requirements of all stakeholders—including the negative ones—must be treated with respect and diligence. Insofar as possible, the requirements of power losers must be incorporated into the new system. A power loser's requirements might look to the power gainers like bells and whistles, but if including such minor features buys peace and cooperation on the project, the trade-off could be worthwhile.
- *Partnership.* Because building partnerships across diverse and often conflicted parts of an organization is the most difficult of our difficult tasks, it's almost always left out of books on requirements methods. *MRP* takes on the subject forthrightly and offers an approach that maximizes the odds of success.

The power shift is what makes software development so hard; if there were no power losers, the process would be, more or less, order-taking and fulfillment.

The Third Surprise: Slow Discovery

Early reveal of system functionality is so 20th century.

When I first began, the charter given to requirements folks was pretty simple: get the spec down quickly in black and white so we can get on with coding. This process was seen more as a reveal than a discovery. We technical guys were supposed to reveal to the poor shlubs who were to inherit the system we'd be building just what the system would do. And their job was to acknowledge—often dazedly—that they'd understood. Once the spec was

What makes system building complex is that the cooperation and participation of power losers is absolutely essential.

done, it was done. Change while we were in the ticklish midst of system installation was unthinkable; we had work to do.

Of course, this approach never worked, but it took us decades to move beyond it. Today, requirements discovery and capture goes on through most of the project. On the best projects, nobody reveals anything, and the very word suggests an authoritarian power imbalance among the parties, something that often leads to dysfunction. Change is afoot from beginning to end. For these reasons, *MRP* suggests the following:

- *Iteration.* The parties work together to evolve requirements that can best satisfy most of their needs. Early visions and models are refined and refined. We aren't sharp enough to get anything right the first time, but we can school ourselves to make successive improvements to our concepts. Relationships are made and matured along the way.
- *Fit criteria.* An important innovation of *MRP* is to turn people's attention slightly away from what they think they need, to concentrate instead on how they would know that a given system did or

did not satisfy their needs. This makes concrete what would otherwise be fogged with abstraction: the work of iterative requirements refinement.


hoc team that establishes requirements has the satisfaction of knowing that it has taken on the lion's share of the project's invention. More important, the relationships that must be a byproduct of

If there were no power losers, the process would be, more or less, order-taking and fulfillment.

Just as important is the book's notion of *misfit criteria*, to deal with things that the system must not do.

In the interest of full disclosure, I need to mention that the book's authors, the Robertsons, are both colleagues and friends. I could hardly have written an unfavorable review of their book, although, of course, I could easily have written no review at all.

I'll end here with a personal reflection. During my time, I've seen requirements work become more complicated, more essential, and at the same time much more rewarding. The diverse ad

this effort are often their own reward. If I treasure the Robertsons' book, it's because its guidance seems most likely to assure the satisfactions and rewards of requirements work. 

TOM DEMARCO is a principal of the Atlantic Systems Guild. Contact him at tdecarlo@systemsguild.com.



Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.

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., Los Alamitos, CA 90720; +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\$56 per year, which includes printed issues plus online access to all issues published since 1984. Go to www.computer.org/subscribe to order and for more information on other subscription prices. Back issues: \$20 for members, \$209.17 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 the accepted version of IEEE-copyrighted material on their own webservers 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. An accepted manuscript is a version which has been revised by the author to incorporate review suggestions, but not the published version with copyediting, proofreading, and formatting added by IEEE. For more information, please go to: http://www.ieee.org/publications_standards/publications/rights/paperversionpolicy.html. 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 © 2013 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.