Alan M. Davis

Predictions and Farewells

Many writers far more eloquent than I have noted that the software industry is in its infancy. What will occur during its next major life phase, adolescence? I do not pretend to be a soothsayer; I hold no magic, no supernatural insight. All I have are my opinions, my fears, and my excitement about the future, which is based on my (what seems to me to be a paltry) 30 years or so of exposure to the software industry. How quickly one moves from being a neophyte to being a geriatric case!

Each fad serves a valuable purpose in shaping industry practices, and each leaves a permanent mark when it departs.

OVERALL

The software industry is ripping itself into segments, each going its own way. These segments are adopting standards and practices so diverse that I expect, by 2010, we will no longer consider "the software industry" to be an industry. The two predominant segments today are the "increasing control of process" (ICP) world and the "decreasing control of process" (DCP) world. Both believe they are on the path to conquering the inherent difficulty of software construction.

The ICP world is mostly on the path toward higher levels of the CMM or some other process improvement path. They believe that by instituting more control on the development process, they'll have more control over the quality of the resulting product. The DCP world employs more and more powerful tools to unburden itself from the tedium and error-proneness of software development. Thus, they program in visual languages to create user interfaces and database applications; they are more interested in achieving higher productivity than measuring it.

SOFTWARE DEVELOPMENT METHODS

Like glaciers, the fads will continue to encroach and retreat, carving great canyons in our terrain. Each serves a valuable purpose in shaping industry practices, and each leaves a permanent mark when it departs. Since joining the industry, I have seen the rise and fall of the Fortran and Cobol eras; I have seen the rise and fall of the Structured era; I have seen the rise and fall of the Case era; and we are all now witnessing the fall of the Object era. Just as happened in earlier eras, the Object era's terms will disappear like glacial ice, but the good ideas created during this era will, like glacially carved canyon walls, persist. I do not believe that such glaciers will cease their cycling for many more decades. The only real question is what name the next age will bear.

REQUIREMENTS

When I wrote *Software Requirements* (Prentice Hall, 1990, 1993) almost a decade ago, the world thought that requirements management was equivalent to model building. This has proven incorrect. As I discussed in my March 1998 editorial, requirements management is about people, about communications, and about attempting to understand before attempting to be understood (*The Seven Habits of Highly Effective People*, Stephen

EDITOR-IN-CHIEF: Alan M. Davis • University of Colorado • adavis@vivaldi.uccs.edu

0740-7459/98/$10.00 © 1998 IEEE

Covey, Simon & Schuster, 1989). I guess I hold little hope in the near term for researchers in requirements management. This fatal proclivity to overlook communications, according to Tom DeMarco in his 1996 keynote at the Second International Conference on Requirements Engineering, is alive and well, and researchers will continue to solve the simple (that is, technical) parts of hard problems.

I have more hopes for practitioners. Over my 20+ years of requirements management consulting, I have seen a strong movement away from having no requirements documents and from formal requirements modeling, and toward

♦ communicating with customers,

♦ writing requirements in natural language, and

♦ storing requirements in a database where they can be annotated, traced, sorted, and filtered.

Design

The lack of a design discipline in software engineering has made us the laughingstock of other engineering disciplines. How pleased I was to see the influx of books over the past five years on the principles of software architecture (*Software Architecture*, Mary Shaw and Dave Garlan, Prentice Hall, 1996) and design patterns (*Design Patterns*, Erich Gamma et al., Addison Wesley Longman, 1995). We are not there yet though, for several reasons:

♦ Practitioners still fail to analyze alternative software architectures before selecting one.

♦ Students of software engineering are rarely exposed to principles of design (there are obviously some exceptions).

♦ The books just scratch the surface when compared to, say, residential or commercial building architecture.

♦ Most importantly, we have not yet produced a discipline of "beauty" and "elegance" in software architecture, let alone the concept of the interplay between form and function, between beauty and utility.

And yet we are clearly making major headway here. I suspect that these problems will go away in a decade or so (see my later comments on academe).

Programming Languages

I predict an end to the language wars by 2020. We are all too obsessed today with language. The visual languages and their associated environments are showing us that we need not select one language. Environments now allow us to program bits and pieces in whatever language makes the most sense for that piece. Those features that make Java relatively platform-independent will become available in most languages. User interfaces and database designs have already become orders of magnitude easier to construct than they were just 10 years ago. These aspects will continue to progress. A few other aspects of programming will likely follow. The result will be less emphasis on algorithms and data structures and more emphasis on thinking at higher levels of abstraction. I recently started programming again after a 15-year hiatus; I am simply overwhelmed by how languages and programming environments have progressed. I have now been freed from details and can think with abstractions from the application domain.

Those features that make Java relatively platform-independent will become available in most languages.

Metrics

The software industry will continue being obsessed with measurement for at least 10.375 more years. Measurement of software product and software process is important. It provides us with yet one more means to assess whether we are getting better or worse as we change. My problem is not with measurement but what we do with data: when we use data to drive decisions at the expense of common sense, we have gone too far. For example, although using a circular saw is 53.2 percent more likely to sever a finger than using a hand saw, that data should not be used to abandon circular saws! In an example closer to home, data indicates that inspections are far more effective at locating defects than is testing; that should be grounds for doing inspections, not for abandoning testing.

Measures should be used to verify the effectiveness of change, not to drive change. We could use a measure to verify that brainstorming was more effective than interviews for requirements elicitation, but trying to achieve better values for such a measure could lead to disastrous results. That is, select a process change that you believe in and monitor its effectiveness by data, but don't let the data drive process changes. In short, keep your brain in gear. There are only a few measurement pundits around today who appreciate the criticality of this difference. Until I see more measurement folks acknowledging this, I will remain pessimistic with respect to their potential for positive effect on industry practices.

Select a process change that you believe in and monitor its effectiveness by data, but don't let the data drive process changes.

ACADEME

I wish I could say I had high hopes for academe, but I do not. My May 1996 editorial expounded on my views that software engineering educators must have real experience in the real world before they attempt to teach the next generation. We have a responsibility to transmit knowledge, and facts, and bases, and reality, not lies and theories that we as academics have learned from earlier generations of inexperienced academics. Yes, there is a place for educating other researchers, but a degree program that purports to teach software engineers (not software engineering researchers) must have educators with extensive industrial experience. It is okay for medical researchers to teach other budding medical researchers. But physicians of tomorrow must be educated by physicians with real patient experience. If software engineering degree programs do not start learning from medical schools and architecture schools (thanks to Colin Potts for pointing out that local practicing architects are often invited into the classroom to comment on the beauty and utility of student designs), we will end up with a new generation of graduates with no ability to produce software.

Computer science programs are in even more danger than software engineering programs. Too much of the core of many computer science degrees hinges upon achieving efficiency of computing resources (time, memory, and so on) that are no longer in short supply, and on building things that most practitioners no longer build. Too little time is devoted to producing reliable, maintainable, quality software.

FAREWELL

This is my time to say good-bye. I am honored to have had the opportunity to serve all of you, *IEEE Software*'s readers, over the span of my tenure as editor-in-chief. I hope that I have made a difference. My goal from the beginning was to do just that: to make the magazine more meaningful and more useful to the practitioner. To share words with you that would help you, the software developer, realize that your profession is an honorable one, that you are fulfilling a cause, but most of all that you have a responsibility to be more than "just" software developers. You have a responsibility to your fellow humans—to make others' lives more meaningful, enjoyable, or satisfying as a result of being touched by either you or your wares. As I have evolved the technical content from research-like articles to the highly practical, I have been happily inundated with letters of support from practitioners. At the same time, I have also received hate mail, mostly from researchers who claim that I have made the magazine less professional, less rigorous, and in their opinion, less helpful. It is impossible to please everybody. I have, however, been true to the goals I put forth when I accepted the honor and the responsibility of this position. Only the to-be-written history books of the Computer Society will judge if my actions were good or bad.

I leave as editor-in-chief with a mission just begun. The transformation of *IEEE Software* is well underway, but it still has many miles to travel. I know of no person more capable or more worthy of serving as the next editor-in-chief than Steve McConnell. He has already made a major difference to those who have been fortunate enough to read his books (*Code Complete*, *Rapid*

Development, and *Software Project Survival Guide*, Microsoft Press) or work with him over the past few years, as I have. As readers of this magazine, all of us have been exposed to gifts from his hand and mind in his monthly Best Practices column.

Good-bye, my friends. I hope to meet many of you during our miscellaneous travels. I thank you, the readers, for your dedication and many letters; without you there would be no magazine. I thank you, the reviewers, for all the assistance you have given the readers and the editorial staff over the years. I thank you, my fellow volunteers on the Editorial Board and Industry Advisory Board, for your vision, for your time, and for your guts; it takes all three of these qualities to effect change. And finally, I thank you, the IEEE Computer Society staff, for the tireless devotion that makes it all possible. ❖

IEEE SOFTWARE

EDITORIAL BOARD

Ted Biggerstaff (Microsoft), Maarten Boasson (Hollandse Signaalapparaten), Terry Bollinger (MITRE), Andy Bytheway (Information Systems Research Centre), David Card (Software Productivity Consortium), Carl Chang (Univ. of Ill., Chicago), Larry Constantine (Constantine & Lockwood), Alan Davis (Univ. of Colorado), Tom DeMarco (The Atlantic Systems Guild), Christof Ebert (Alcatel Telecom), Ophir Frieder (Florida Inst. of Technology), Robert Glass (Computing Trends), Sadahiro Isoda (Toyohashi Univ. of Technology), Natalia Juristo (Universidad Politécnica de Madrid), Barbara Kitchenham (Univ. of Keele), Tomoo Matsubara (Matsubara Consulting), Steve McConnell (Construx Software Builders), Nancy Mead (Software Eng. Inst.), Stephen Mellor (Project Technology), Tsuyoshi Nakajima (Mitsubishi), Roger Pressman (RS Pressman & Associates), Chandra Shekaran (Microsoft), Pradip Srimani (Colorado State Univ.), Wolfgang Strigel (Software Productivity Centre)

INDUSTRY ADVISORY BOARD

Stephen Andriole (Safeguard Scientifics/TL Ventures), Robert Cochran (Catalyst Software), Annie Kuntzmann-Combelles (Objectif Technologie), Enrique Draier (Netsystem SA), Kenneth Dymond (Process Inc US), William Griffin (GTE Labs), Arthur Hersh (Hersh Group), Eric Horvitz (Microsoft), Takaya Ishida (Mitsubishi), Dehua Ju (ASTI Shanghai), Donna Kasperson (Unisys), Günter Koch (Austrian Research Centers), Wojtek Kozaczynski (System Software Associates), Karen Mackey (Lockheed Martin), Masao Matsumoto (Univ. of Tsukuba), Susan Mickel (Rational Univ.), Melissa Murphy (Sandia), Kiyoh Nakamura (Fujitsu), Grant Rule (Guild of Independent Function Point Analysts), Martyn Thomas (Praxis), Sadakazu Watanabe (Toshiba)

CONTRIBUTING EDITORS

Ware Myers, Shari Lawrence Pfleeger, Ellen Ullman, Mike Yacci

MAGAZINE OPERATIONS COMMITTEE

Carl Chang (chair), William Everett (vice chair), Gul Agha, George Cybenko, Alan M. Davis, Stephen L. Diamond, William Grosky, Bertram Herzog, Daniel E. O'Leary, Edward A. Parrish, Dave Pessel, Charles Petrie, Michael R. Williams, Yervant Zorian

PUBLICATIONS BOARD

Benjamin Wah (chair), Jon Butler, Carl Chang, Alan Clements, Dante Del Corso, Richard Eckhouse, William Everett, Francis Lau, Sorel Reisman

EDITOR-IN-CHIEF: **ALAN M. DAVIS**
10662 LOS VAQUEROS CIRCLE
PO BOX 3014
LOS ALAMITOS, CA 90720-1314
software@computer.org

EDITOR-IN-CHIEF EMERITUS
CARL CHANG

ASSOCIATE EDITORS-IN-CHIEF:
TED BIGGERSTAFF, LARRY CONSTANTINE, TOMOO MATSUBARA

MANAGING EDITOR: **DALE C. STROK**
dstrok@computer.org
STAFF EDITORS: **JAMES SANDERS, ANNE LEAR**

DESIGN DIRECTOR: **JILL BOYER**
COVER ILLUSTRATION: **DIRK HAGNER**
TECHNICAL ILLUSTRATOR: **ALEX TORRES**
PRODUCTION ARTIST: **JILL BOYER**

PUBLISHER: **MATT LOEB**
MEMBERSHIP/CIRCULATION
MARKETING MANAGER: **GEORGANN CARTER**
ADVERTISING MANAGER: **PATRICIA GARVEY**
ADVERTISING COORDINATOR:
MARIAN ANDERSON
MANUSCRIPT ASSISTANT: **ANGIE SU**
software@computer.org

BPA INTERNATIONAL ®