as I write this issue's From the Editor column, I have just returned from our annual board meeting. At this meeting, the magazine’s combined Editorial and Industrial Advisory Boards meet to reflect on the past year and plan for the coming year. This is what makes IEEE Software unique among the computing literature.

**Topically reactive publications**

Most technical computing publications are topically reactive—they depend mainly on unsolicited submissions to fill their pages every year. Prior to joining IEEE Software, I was editor in chief of another academic journal for almost 10 years. Readers would often suggest I publish more on this or that topic. Unfortunately, because the topics we published were what the contributors chose to submit, we had little flexibility in “engineering” the journal’s content.

For the most part, leading a reactive publication is much like riding an inner tube over a waterfall. Rather than planning your trip, you’re along for the ride. On the other hand, reactive publications are useful in that they accurately reflect the topics that researchers and other luminaries are currently thinking about. For instance, if lots of people are thinking about inspections, you’ll get lots of inspection papers. If lots of people are working on glass box testing methods, you’ll get lots of glass box testing papers. Because the goals of many such publications are to document an idea’s source and facilitate communication among a community’s researchers, this actually works the way it is intended.

**Topically proactive publications**

On the other hand, a publication such as IEEE Software has a unique goal: to expose practitioners to important trends in the software development community. This goal lets us be topically proactive. In other words, rather than simply depending on whatever gets submitted, we can engineer submissions through a well-planned and organized focus-issue mechanism. By proactively selecting focus-issue topics rather than waiting for what comes in, we can plan our “trip.” This isn’t to say that we don’t encourage unsolicited “regular” contributions. In fact, we welcome them. However, the focus-issue mechanism helps us set IEEE Software’s direction.

But how do we decide what topics warrant a focus issue? This is where our annual board meeting comes in. Every summer, our Editorial and Industrial Advisory Boards meet to discuss important trends in the software development industry. The bulk of the three-day meeting entails discussions of trends and topics about which software practitioners need to know. By the final day, we usually have identified dozens of topics. Here are just some of the topics we came up with this June:
IEEE Software

FROM THE EDITOR

- The effect of software development practices on legal issues. For example, to what degree do developers’ software engineering practices affect software product liability?
- Metaphors for software development. What are some of the metaphors other than engineering that have been suggested for software development? Does theatre or agriculture work? (For example, see “Beyond Requirements: Software Making as Art” by Rob Austin and Lee Devin in the Jan./Feb. 2003 issue of IEEE Software.)
- Email-based work environments. How can we handle and exploit environments whose communications are heavily automated?
- Resource-aware computing. How can software be written to dynamically manage and optimize resources such as power and bandwidth?
- The “ilities.” This topic seems to come up almost every year. We are all aware of software’s “ility” constraints: maintainability, reliability, and so on.
- The impact of market forces on software engineering practices. This includes access to cheaper labor.
- Better understanding of software engineering by business managers, and vice versa. How can we get managers to better understand software development? How can we get software developers to better understand management issues?
- Career management. How can software developers reinvent themselves in response to the rapid changes our field is undergoing in both its technology and its organizations? What impact has the automation of so many software development tasks had on people’s careers?
- Career comparisons across cultures and continents. What does a software developer’s career in various parts of the world look like?
- Pros and cons of SWEBOK. What significance will the Engineering Body of Knowledge project have on the practicing software development community? What are its ramifications?
- Characteristics of future programmers. Tomorrow’s programmers will have been raised on high-speed connections and peer-to-peer resource sharing. Will they differ from today’s programmers?
- Software engineering education is not training. Fundamental principles distinguish the two. What should software engineering education not teach?
- New user interfaces. How are developers meeting the need for today’s software to operate with a variety of different input devices, from cell phones to full-screen Web browsers?
- The growth and integration of interoperable, heterogeneous systems. As the number of systems with which the “typical” system must interact increases, how do we keep track? Are these “systems of systems” or just “systems”?
- Design and architecture for security. How can we build security into the architecture and design of systems?
- Planning and building for evolvability. The old saying is “plan for change.” What effect does evolvability have on our systems? How can our designs and architecture support evolvability?
- Commercial off-the-shelf system legacy issues. What kinds of legacy issues are COTS integrators encountering as systems age, vendors go out of business, and product support is phased out?
- Obfuscation of automation. Is automation making developers and users lose sight of what is being done with issues such as security and correctness on their behalf?
- Safe programming by nonprogrammers. Can nonprogrammers program safely? If not, can anything be done about it?
- Truisms, “urban” legends of software engineering. We’ve all heard them and probably used them in an example or in making a point. For example, did a jet fighter flip upside down every time it crossed the equator? Does the cost of repairing an
error increase by a factor of 10 for every additional phase to which it leaks before being discovered? Does adding people to a late project just make it later? Who said these? Can they be documented? In what context was the original statement made?

Borderless computing. It is now possible for a software system to have its database hosted in Greece, its interface and client-side processing software retrieved from a Web server in Mexico, and the server application executed on a computer in the Ukraine. What effect does geographically dispersing a software system have on development?

Technology transfer. Every year, academics and scientists in research laboratories make dozens if not hundreds of important new discoveries (often funded by industry) about software development. At the same time, software practitioners encounter hundreds if not thousands of problems that these new discoveries could solve. How can technology transfer be facilitated? Do any particular models work?

Paring it down

Unfortunately, we only have six issues each year. So our combined Editorial and Industrial Advisory Boards spend the meeting’s third day reducing our dozens of potential special topics into a short list that we think are most relevant to Software’s readers. While current readers are of course a high priority, the leading topics are also influenced by what we think the software development community as a whole should be thinking about, if they aren’t already.

Here is our short list (see the earlier list for longer descriptions):

- Software practice and legal issues
- Metaphors for software development
- How to handle and exploit electronic-based environments
- The “-ilities”
- Commercial off-the-shelf and legacy issues
- Better understanding of software engineering by business managers and vice versa
- International career comparison
- Borderless computing
- Resource-aware computing
- New user interfaces
- Software engineering education

Making it happen

Once we’ve identified potential topics for focus issues, board members take responsibility for finding people to coordinate them as well as to write and review articles. Of course, not every topic will pan out—we might not find the right champions and guest editors. Moreover, market or world events might bring other topics to prominence. And we might get timely unsolicited special-issue proposals that we think take precedence over our short list.

During the next few months, you will probably see calls for contributions to focus issues related to these topics. If one of these topics interests you, be proactive and contact us.

How did we do this year? One of the roles of IEEE Software’s Editorial and Industrial Advisory Boards is to help identify topics our readers are interested in. We are all vitally interested in how well we’re doing at this. Did we omit some topics we should have selected? Did we miss some important issues completely? Please let us know by emailing me at warren.harrison@computer.org.

FROM THE EDITOR

Next Issue: The State of the Practice of Software Engineering

EDITOR IN CHIEF: Warren Harrison
10662 Los Vaqueros Circle
Los Alamitos, CA 90720-1014
warren.harrison@computer.org

EDITOR IN CHIEF EMERITUS: Steve McConnell, Construx Software

ASSOCIATE EDITORS IN CHIEF

Education and Training: Don Bagert, Rose-Hulman Inst. of Technology; don.bagert@rose-hulman.edu
Requirements: Christof Ebert, Alcatel christof.ebert@alcatel.com
Management: Ann Miller, University of Missouri, Rolla millera@ece.umr.edu
Quality: Jeffrey Voas, Cigital voas@cigital.com
Experience Reports: Wolfgang Strigel, Software Productivity Center; strigel@spc.ca

EDITORIAL BOARD

Nancy Eickelmann, Motorola Labs
Richard Fairley, Oregon Graduate Institute
Robert Glass, Compaq
Jane Hayes, University of Kentucky
Andy Hunt, Pragmatic Programmers
Warren Keuffel, independent consultant
Brian Lawrence, Coyote Valley Software
Karen Mackey, Discysystems
Dipendra Moitra, Infosys Technologies, India
Don Reifer, Reifer Consultants
Suzanne Robertson, Atlantic Systems Guild
Richard H. Thyen, Calif. State Univ. Sacramento
Dave Thomas, Pragmatic Programmers

INDUSTRY ADVISORY BOARD

Robert Cochran, Catalyst Software (chair)
Dave Aucsmith, Microsoft
Annie Kuntzmann-Combelles, Q-Labs
David Dorenbos, Motorola Labs
Enrique Draier, MAPA LatinAmerica
David Hsiao, Cisco Systems
Takaya Ishida, Mitsubishi Electric Corp.
Dehua Ju, ASTI Shanghai
Donna Kaspersen, Science Applications International
Paweł Knifflic, Hermes SoftLabs
Wojtek Kozaczynski, Microsoft
Tomoo Matusbara, Matusbara Consulting
Masao Matsumoto, Univ. of Tsukuba
Dorothy McKinney, Lockheed Martin Space Systems
Stephen Mellor, Project Technology
Susan Mickel, AgileTV
Dave Moore, Vulcan Northwest
Melissa Murphy, Sandia National Laboratories
Grant Rule, Software Measurement Services
Girish Seshagiri, Advanced Information Services
Chandra Shekaran, Microsoft
Martyn Thomas, Praxis
Rob Thomsett, The Thomsett Company
John Vu, The Boeing Company
Simon Wright, Integrated Chipware
Tsuneo Yamaura, Hitachi Software Engineering

MAGAZINE OPERATIONS COMMITTEE


PUBLICATIONS BOARD

Rangachar Kasturi (chair), Jean Bacon, Lakmi Bhuyan, Mark Christensen, Thomas Keefe, Dipendra Moitra, Steven L. Tanimoto, Anand Tripathi

September/October 2003 IEEE SOFTWARE 7