The Only Constant Is Change

Forrest Shull

WHAT TOPICS WILL software developers be talking about a year from now?

The fast-changing nature of our field is one of the things that make working in software so much fun—and so challenging. Grady Booch often writes in these pages about the way in which software has grown to underlie so much of our modern world. Michiel van Genuchten and Les Hatton in their Impact column describe how software has penetrated so many of the systems and services that we as consumers often take for granted—airliners, mass transit, and washing machines—where it lies hidden in plain sight. Meanwhile, many of our other authors frequently write about the enabling technologies that have allowed software to become so influential—the methods, tools, and practices that enable software developers to produce quality software that becomes embedded all over our world.

The joy and the challenge of my job as editor in chief is staying abreast of a broad swath of contemporary research and practice to make sure that we’re featuring the right methods, tools, and practices in this magazine, the technologies that are relevant, important, and helpful to practicing software engineers in a variety of roles and across many different domains.

Some of the topics that I feel most strongly about have already been turned into special issues. For example, next-generation mobile software and new developments in programming languages will be featured soon, while software analytics can be found in the current issue. But looking further ahead is always important. One of the ways that I do this is with the help of our editorial and advisory board members. Members are invited to the boards based on their knowledge and visibility in the software engineering field, as well as their willingness and ability to contribute to the magazine. I manage the board composition to make sure that we have coverage of important industries, viewpoints, and types of software, so that the advice I get is as comprehensive as possible and representative of concerns from across our readership.

Once a year, the boards and I come together so that I can hear what’s on the minds of experts in various sectors of the industry and to ensure that our content is on target going forward. We brainstorm as many different topics of interest as possible (and in a dynamic field such as ours, it’s probably no surprise that we easily reach 50 or more topics), with all of us ask-
SOFTWARE EXPERTS SUMMIT 2013

Speaking of hot topics, big data analytics clearly continues to be high on almost everyone’s list. For that reason, we dedicated our annual Software Experts Summit to an investigation of various facets of that issue. This year, our agenda featured a range of compelling speakers who discussed the latest developments and research. We gave our speakers the challenge to discuss “smart data,” meaning that rather than focus on the mechanics of dealing with increasingly large amounts of data, we asked them to examine how users get accurate meaning and actionable insights from real-world data, coming as it often does from multiple sources, formats, and media.

Videos and presentations from our speakers will be placed on the event website, www.computer.org/ses13, as they become available. Our lineup this year included keynotes from Paul Zikopoulos (IBM) on how organizations can get real-time, actionable insights from the data encoded in social media and other sources, and James Whittaker (Microsoft), who shared ongoing work that processes data to allow computer interfaces to understand and respond to the intention behind a user’s interactions.

Additional technical talks included Ayse Bener (Ryerson University) discussing how sampling can be used to find smaller datasets more tractable to analysis and providing robust insights; Wolfgang Schulte (Microsoft) on how data analytics is applied to software development data at Microsoft to support evidence-based decision making; John Howie (Cloud Security Alliance), who looked at the trade-offs and risks involved with privacy in big data; and Jeromy Carriere (Google) who described the infrastructural advances that have allowed the practical application of big data analytics.

Finally, a panel led by Hakan Erdogmus (Kalemun Research) kicked off a reflection on the day’s content with experts from diverse application domains such as defense, medical, and outsourcing, then invited the audience members to add their own perspectives.

SES is a public event that brings together some of the key thought leaders associated with IEEE Software (board members, authors, interviewees, and guest editors) as well as local experts to tackle a problem of compelling interest. We select expert speakers who clearly communicate the technical issues on which they work and design the day’s agenda to allow for interactivity and networking. We’re already planning SES 2014, so watch this space for future announcements!
From the editor

my columns, this installment is going to present more questions than answers—but I find the insight into the worries that are on thought leaders’ minds is an important source of information, and I hope you will too.

Refactoring/Reengineering

Although the practice of refactoring is hardly new, it seems to be taking on additional relevance these days. One reason for this is the increasing interest in technical debt, which is refocusing many developers on the questions of how and when to refactor their software so as to deliver the most business value over the lifetime of a project and make the right technical trade-offs.

As with many of the other topics that the boards and I came up with, refactoring becomes more difficult and raises many technical issues that aren’t well understood when applied in practice to the types of systems that developers work with today. Large-scale and high-complexity systems are difficult enough, to be sure, but polyglot and integrated systems make the trade-offs and “optimal” outcomes even harder to parse. Also relevant to this discus-

SOFTWARE ENGINEERING IN PRACTICE AWARD

IEEE Software continues its policy of sponsoring best paper awards at major conferences. It does this both to recognize the best work going on in our field and to call extra attention to the types of content that we, its editors, particularly prize: practical work with concrete applications for software developers that is also rigorously analyzed and reported.

The International Conference on Software Engineering (ICSE) is one of the flagship conferences for software engineering researchers, and we have a strong affinity for the “Software Engineering in Practice” (SEiP) track, which highlights work being done to address real and current software development problems. The SEiP goal of fostering more communication between researchers and software developers, and helping communicate state-of-the-practice case studies, is one that resonates well. Because this year’s ICSE was held in Silicon Valley, in close proximity to some of the great hotbeds of software innovation and development, the SEiP track was quite substantial and robust. Several of the events, including keynotes from industry and a panel on technical debt, were standing room only. For more information about the unique aspects of the SEiP track, and our award there, see my short interview with track cochair (and IEEE Software editorial board member) Jane Cleland-Huang at www.computer.org/software-multimedia.

Our award this year went to a paper that contained a wealth of experience on finding and fixing bugs that are characteristic to big data systems. The authors of the winning paper, “Characteristic Study on Failures of Production Distributed Data-Parallel Programs,” are Sihan Li, Hucheng Zhou, Haoxiang Lin, Tian Xiao, Haibo Lin, Wei Lin, and Tao Xie.

As always with our best paper awards, a short list of nominated papers was created by the track chairs and a selection committee drawn from the track organizers and our magazine boards. This year, our selection committee consisted of Ipek Ozkaya, Frances Paulisch, Bob Hall, and Erik Simmons. I’d like to again say congratulations to our award winners and thanks to our selection committee!
From the editor

From the editor

From the editor

From the editor

From the editor

From the editor

From the editor

From the editor

From the editor

From the editor

A Fresh Look at the Science of Measurement

In grad school, I once took a class with a professor of artificial intelligence (AI) who said that, in his view at least, AI too often got an undeservedly bad reputation as a research field. Anytime an effective solution was developed for a particular problem, it was spun off and became its own field, while the things left under the umbrella of AI were the hardest and most intractable problems remaining in the realm of basic research. I often feel that the same has happened to software metrics: as an abstract field, some critics look at metrics as an unexciting area that can be divorced from their needs. And yet, metrics that have proven successful are woven into day-to-day practice: any project has to do some estimation of schedule and cost (and an array of techniques exist that are both usable and useful), agile teams are often skilled with using measures of velocity to steer project decisions, and many organizations have their own toolsets that provide useful measurement-based insights.

Given all of the exciting work going on in software analytics, this seems an opportune time to revisit measurement science and take a new look at where metrics have already successfully embedded themselves into software practice, becoming so intuitive that we rarely notice them. It would also be interesting to examine the extent to which we can make truly evidence-based decisions during software development to maximize our chances of success. The tension between collecting nonintrusive metrics and meaningful metrics, as always, is relevant here. Technical debt, another topic of an earlier special issue, is also highly related as the research community and tool vendors increasingly analyze code and process metrics to detect debt that might be accumulating unintentionally and to provide triggers to alert software developers when debt might be getting out of hand.
From the editor

Ongoing challenges include the application of metrics to contemporary technologies and at contemporary scales—for example, how accurate can we be in estimating and monitoring performance in the cloud?

Designing the User Experience
As software and systems become more ubiquitous, users have to deal with ever-larger amounts of information and a multitude of capabilities. Poorly designed interfaces and poorly integrated systems lead users to unpleasant experiences with technology or “IT overload,” which prevent effective use of available options. A constellation of related questions comes up around how to address this problem:

Whose job is it to consider the user experience during software design—and what tools can be leveraged to help do it well? How do you train developers to design software effectively? Can software help address the problem itself—for example, by having interfaces with some intelligence behind them?

As if all of that wasn’t complicated enough, can we reason about the composability of the user experience—that is, if we know something about component usability, can we reason about how easy or cumbersome a user will find it to work with the whole system? And how do requirements engineers and software developers handle unusual (but increasingly common) nonfunctional requirements related to the user experience, such as a computer game that should be “fun” or “pleasurable?”

The Effective Software Manager
Not that anyone has recently stopped worrying about how to be an effective manager of software projects, but given recent trends, this might be a particularly good time to bring together some thoughts on what it means to manage well under current conditions. A particularly interesting question is what effective strategies exist for combining bottom-up empowerment with top-down management, which is certainly related to the question of how to manage effectively in an agile context. But we need not be...
limited to agile teams: What recently emerged best practices do managers use on long-lived, mission- or safety-critical projects to safeguard users as well as the large investment in cost?

A recent line of research within software engineering focuses on making sure that teams’ technical decisions align with business value—but how well are the resulting methods working in practice, and are they making their way into the hands of practicing managers? How do managers deal with multicultural issues on large, geographically distributed project teams?

As I mentioned, these are just a few of the priorities we identified, but I’d be happy to hear from you, our readers, if you have hot topics of your own or want to react to the ideas here. Contact me at fshull@computer.org.

Many thanks to our board members who participated in this year’s discussion: Ayse Bener (Ryerson University), Anita Carleton (Software Engineering Institute), Hakan Erdogmus (Kalemun Research), Magnus Larsson (ABB), Steve McConnell (Construx Software), Grigori Melnik (Microsoft), Ipek Ozkaya (Software Engineering Institute), Rafael Prikladnicki (Pontifical Catholic University of Rio Grande do Sul), Linda Rising (independent consultant), Wolfgang Strigel (independent consultant), Girish Suryanarayana (Siemens), Michiel van Genuchten (mt:onyx), Adam Welc (Oracle Labs), James Whittaker (Microsoft), Rebecca Wirfs-Brock (Wirfs-Brock Associates), and Olaf Zimmermann (University of Applied Sciences, Rapperswil, Switzerland).

FORREST SHULL is a division director at the Fraunhofer Center for Experimental Software Engineering in Maryland, a nonprofit research and tech transfer organization, where he leads the Measurement and Knowledge Management Division. He’s also an adjunct professor at the University of Maryland College Park and editor in chief of IEEE Software. Contact him at fshull@computer.org.

See www.computer.org/software-multimedia for multimedia content related to this article.