I’VE SUBSCRIBED to IEEE Software since it started in 1984. I’ve always liked its practical focus and that it covers all types of software from embedded systems to large business systems. It was, and maybe still is, the most readable and useful of the magazines trying to bridge the divide between researchers and practitioners.

For the past few years, though, I haven’t found much I want to read in the magazine. Typically, I skim the magazine, reading some of the columns. Occasionally, a longer article catches my eye, but the average number of those articles that I read is significantly less than one per issue. As I get older and more cynical about “advances” in the discipline, I’m less inclined to read about unvalidated research that has no obvious relevance to me.

In the July/August 2015 issue, I read Philippe Kruchten’s column on lifelong learning and the importance of professional development.1 This really brought home why I was disenchanted with the magazine. For me at least, hardly any of the articles in recent issues have been useful for professional development.

What were most valuable were articles about areas of software engineering outside my expertise.

This is a significant change from a few years ago. The magazine used to have many useful, general articles on software engineering that I learned from and recommended to my students. But now, most articles seem specialized or are short columns presenting opinions and one person’s experience. Columns are all very well, but their short length means they’re tasters, not a basis for professional development.

A Tale of Two Issues
I looked back 20 years to the July 1995 issue, which offered three articles on general topics that I recall reading:

• “Seven More Myths of Formal Methods” discussed misconceptions people had (and I suspect still have) about using formal methods in software engineering.2
• “Case Studies for Method and Tool Evaluation” explained the benefits and pitfalls of evaluation using case studies.3
• “Contemporary Application-Domain Taxonomies” discussed the importance of domain-specific software engineering and presented ways to organize and understand those domains.4

If you compare the number of citations of these articles (as shown in the IEEE Xplore Digital Library) with that of the “research” articles in that issue, the difference is marked. The research
articles have single-figure citations; the more general articles have 49, 107, and 21 citations, respectively.

In contrast, consider the July/August 2015 issue. All but one of the longer articles were on specialized research topics, and the one article that claimed to be a tutorial was actually a summary of a longer research paper published in 2014. It wasn’t really a tutorial because it lacked information on what the reader needed to know to benefit from the article and guidance on how to learn more about the topic.

That issue represented a missed opportunity. The focus was security and privacy on the Web, and a good tutorial on this important topic would have been immensely valuable.

The Three Phases of Learning
To become more relevant to both researchers and practitioners, IEEE Software should refocus and become the first place software engineers turn to for professional development in software engineering and related areas.

To understand what’s needed to effectively support professional development, we need to understand what’s involved in self-learning. I’ve spent more than 40 years learning about software engineering since I was a physicist in the 1970s. I have no idea how typical I am, but for me, learning has three interleaved phases:

1. The what-and-why phase involves discovering an area that’s new to me and understanding why I might want to know more about it.
2. The making-sense phase involves starting to learn about this new area and making sense of its concepts, principles, and processes.
3. The deeper-exploration phase involves exploring an area in more depth by looking at more specialized resources, practice reports, case studies, and so on. Critically, I want to know when a technology isn’t useful as well as how to use it when it is.

When you’re learning about a new area, research papers aren’t of much interest. You don’t know enough to assess their significance and relevance, so you can’t fit them into the knowledge framework you’re building. Even in the deeper-exploration phase, research articles are relevant only when they discuss the relationships between the reported research and everyday practice.

To suggest what would be useful in each of these phases, I’ll use the example of the July/August 2015 issue’s focus (security and privacy on the Web).

For the what-and-why phase, the issue could have featured an overview of Web security and privacy issues and why they’re relevant to all software engineers, not just those who are developing Web applications. The guest editors’ introduction in the issue covered some of this but wasn’t really for readers with no background in the area.

For the making-sense phase, a tutorial could have introduced Web security and privacy and the critical things you need to know about the area.

For the deeper-exploration phase, a survey could have covered Web security and privacy resources such as books, videos, and white papers. It would have explained what security and privacy aspects they cover and their strengths and weaknesses.

Rethinking the Magazine
So, I believe IEEE Software should rethink the type and content of articles it publishes. I’d like to see the following.

Tutorials
In Kruchten’s article, he asked, “What do I know about the MEAN stack?” Well, I didn’t know anything about it (I discovered that it means MongoDB, Express, AngularJS, and Node.js), but I’d like to read a well-written tutorial about it. Tutorials used to be more common in the magazine; it’s a pity they’ve mostly disappeared.

Reviews
Reviews of tools and technologies are really helpful for people learning about them. So, to continue the MEAN stack example, I’d like to see a review of NoSQL databases (including MongoDB), along with guidance on when to use them.
Practice-Focused Research
Most research articles are pretty useless to practitioners because they don’t relate the reported research to practice. They often provide tentative, preliminary conclusions rather than useful lessons learned and solid information on how to do things better. It makes sense to continue to publish articles on recent research, but only if they explicitly discuss lessons learned, what went wrong as well as what went right, and how the research is relevant to practice.

Starting a Conversation
Learning can be a lonely business, and it’s sometimes useful to know about the problems others have faced and their opinions on what’s good and not so good about learning resources. So, articles shouldn’t simply be monologues; they should be conversations between authors and readers.

Currently, this doesn’t seem possible with IEEE Software articles. Readers can’t comment on articles, find out what others think of an article, or offer opinions differing from the authors’. To provide commenting will require a 21st-century publishing platform, but surely this isn’t too much to ask in 2015.

How to Get There
I know from my experience as a journal editor that if you simply rely on submitted articles, you won’t necessarily get the kind of articles you want. Researchers submit articles to IEEE Software because they get tenure and promotion benefits. Writing the kind of articles I’m suggesting here provides few career benefits for ambitious researchers.

Realistically, the changes to the magazine that I think are needed won’t happen without the proactive commissioning of articles. The best way to encourage submission of good professional-development articles might be to pay for them. I don’t think this means paying commercial consultancy rates, but writing well takes time. It’s fair to offer some reward to those willing to share their knowledge. Paying $1,000 for a well-written tutorial would be money well spent.

I don’t believe this is financially unrealistic. Membership in the IEEE Computer Society and access to its publications offer good value. Most members wouldn’t object to an increase in their dues if they could see tangible professional-development benefits.

Lifelong learning is critical to our discipline’s future. We need to be able to leverage the wisdom of experienced engineers in new areas and address the worldwide skills shortage of software engineers. As a long-term reader, I want to stay loyal to IEEE Software, so I’d like to see a positive commitment to change and to taking the lead in this important area.

References

IAN SOMMERVILLE is an author and a consultant interested in systems of systems and the relationships between software and systems engineering. He’s the author of a widely used software engineering textbook. Contact him at software-engineering@iansommerville.com.