Seeking Your Insights

Cesare Pautasso and Olaf Zimmermann

HOW CAN WE keep knowledge from evaporating? Wouldn't it be nice if your valuable experience earned in one project could be exchanged with everyone? What you learned the hard way would become easily accessible to others (and vice versa). Short stories and longer experience reports, reflections and retrospectives, as well as patterns and styles all attempt to gather useful reusable knowledge nuggets that collectively make up the state of the software practice.

Making a good decision is hard. Personal experience can be a good source of evidence to back your decisions, but this is usually limited because you can't experience everything yourself. So, you rely on trusted sources of experience—for example, a colleague, another member of your professional network, or a well-known authority in the field. Additionally, you can attend a reputable conference, search through experience-sharing sites (for example, InfoQ, www.infoq.com, and Stack Overflow, http://stackoverflow.com), and even read magazines (as you're doing now). Written sources have the advantage that the shared experience is "aged." Writing stuff down forces you to reflect on your decision—successful sharing implies hardening—so publishing written knowledge is beneficial for both sides.

This Insights column is one place to write up knowledge nuggets. We're grateful to *IEEE Software* for the opportunity to continue along the path that Linda Rising paved to give a voice to busy software professionals and let their stories be heard. This column's goal remains unchanged—share real-world experience and take a snapshot of where practical software engineering has been, is now, and is heading.

The *IEEE Software* legacy includes foundational and influential articles such as "Reverse Engineering and Design Recovery: A Taxonomy," "Visualizing the Performance of Parallel Programs," "Who Needs an Architect?," "The 4+1 View of Architecture," and "Global Software Development," and other frequently cited ones.[1] Articles in a magazine such as *IEEE Software* are peer reviewed and professionally edited. They can be viewed as a trusted, curated source of experience. And, like conferences, they might provide just enough serendipity so that you can find insights into topics you normally wouldn't look into.

What We're Looking For

The knowledge we're looking for falls into these broad categories:

- patterns of all kinds,
- contemporary architectural styles,
- proven methods and techniques, and
- emerging technologies and tools.

We're interested in hearing about both your positive and negative experiences applying these categories in a given context with reproducible effects. We greatly appreciate critiques, actionable comments, and constructive advice.

Your contributions should be more mature and refined than what you would normally find on most blogs, but just as focused, timely, and relevant. The advantage? Your insights will be presented to the broad *IEEE Software* readership and preserved as part of the IEEE Xplore digital library (http://ieeexplore.ieee.org).

Getting Started

To help you get started, the following questions are intended to provoke a reaction that should lead you to the insights we seek. We ask these questions regularly when performing software reviews. Also, insightful answers to these questions in the context of similar projects or from trusted references have helped us when we worked on industry projects.

We've structured the questions roughly following the basic software engineering life cycle. You don't need to answer them all—it's okay if you pick a few, as long as your answers are substantial enough to be relevant to our readers.

THE CONTEXT OCTOPUS

Philippe Kruchten's octopus-and-frog metaphor lists eight dimensions of context: (system) size, (system) criticality, system age, team distribution, rate of change, preexistence of a stable architecture, governance (including management rules), and business model (internal system, commercial product, or open source software).[1] These dimensions are worth knowing and disclosing when it comes to experience sharing—one-size-fits-all doesn't work in software engineering.

Reference

1. P. Kruchten, "Contextualizing Agile Software Development," *J. Software Evolution and Process*, vol. 25, no. 4, 2013, pp. 351–361.

Context and Problem

What kind of software project are you describing? In what context did it take place, in terms of the eight dimensions of Philippe Kruchten's context octopus (see the sidebar)?

What problem were you trying to solve? How closely did you involve your problem's stakeholders in the project? How did you deal with their feedback and changing requirements?

Project Management

How did you estimate, manage, and mitigate the technical risk? How much technical debt did you accumulate, and how did you deal with it?

Did you follow some particular software engineering method, set of methods, or practices? For example, would you consider your project agile?

Architecture Design

What were your three most relevant architectural decisions? How did you find and evaluate design alternatives (solution options) for them? How did you pick one? Are you still content with your decisions?

What's your experience with particular modeling notations in real-world projects? Which parts of the system did you model and why? Did this pay off?

Development and Test

How did you test that functional and nonfunctional requirements were satisfied? Did you also try to prove this?

How long was your release cycle? How was your experience with continuous integration, test automation, and software configuration management (versioning)?

Operations and Maintenance

What was your approach to IT service or systems management? Did the chosen frameworks, libraries, and tools deliver on their promises?

If you applied a DevOps (development operations) approach, how did you benefit from it in the short and long terms?

Reflection

When did the constructed system go live? How did it live up to your expectations? How did it survive against the actual workload, and how did you evolve it in response to a growing workload?

In retrospect, what went well and what didn't? Could you solve all the problems (on time and within budget)? What will you do differently next time?

Examples of Insights

Here are some examples of insights we're looking for.

In software development, as in real estate, location matters.[2] Meaningful communication between team members decreases as distance increases. This holds for teams working with people located around the world but also affects the office layout of colocated teams.

Large organizations have difficulty enforcing compliance with technical standards because top-down communication might not be sufficient to get everyone on board.[3] This isn't necessarily a problem of convincing people to commit, but of simply making them aware of the decisions affecting them. As opposed to relying on centralized document archives, which must be continuously searched for relevant information, it helps to push the knowledge directly to the intended recipients when they need it to perform their tasks.

Software architects can learn something from meteorologists.[4] In the same way it's important to forecast an incoming hurricane's path, agile software architects need to anticipate future changes in their design and forecast how the software system will likely evolve.

Given the shortage of skilled IT personnel, it might pay to look for talent outside traditional pools.[5] Not only electronics engineers or graduates in math and physics, but also people with a creative-arts background might show the out-of-the-box thinking and problem-solving skills a software development position requires. Additionally, the latter folk might be more motivated to keep sharpening their skills on the job.

When you're migrating a legacy system to a new architecture—for example, to turn it into a cloud native application—it helps to have the migration team include the following two roles.[6] The *system steward* knows about the existing system and ensures that the pace of the migration doesn't break it. In contrast, the *future visionary* is completely immersed in the latest technologies and helps push the project toward the endgame by stretching the team's knowledge and abilities.

These are all concrete, actionable pieces of knowledge you can share without compromising your organization's intellectual property. Such knowledge pieces emerge from all industry sectors and business domains and might cover one or more software engineering life-cycle phases. It should be possible to try out your insights and experiment with them in a new project or a well-established one that seeks some improvement. What we're not looking for are buzzword-filled marketing pitches or short-lived, product-specific experiences.

To get your insights presented to our readers, we offer an open ear and a patient hand to coach you and shepherd your drafts. Between us, we share more than 30 years of academic and industry experience in collecting, shaping, and revising technical prose and advising, guiding, and mentoring authors. We're committed to achieve a quick turnaround with feedback about proposals with your initial ideas. You'll also get professional support for editing your story as it goes through the publishing process.

Send in your best insights. We'll be pleased if you can convince your friends and colleagues to share their stories, experience, and knowledge with us and everyone else.

ABOUT THE AUTHORS

CESARE PAUTASSO is an associate professor of informatics at the University of Lugano. His research group focuses on building experimental systems to explore the architecture, design, and engineering of next-generation Web information systems. Previously, he was a researcher at the IBM Zurich Research Lab and a senior researcher at ETH Zurich. His teaching, training, and consulting activities, spanning both industry and academia, cover advanced topics related to emerging Web technologies, RESTful business process management, and cloud computing. He's a coauthor of *SOA with REST: Principles, Patterns & Constraints for Building Enterprise Solutions with REST* (Prentice Hall, 2012). Pautasso received a PhD in computer science from ETH Zurich. He was the program cochair of ICSOC (Int'l Conf. on Service-Oriented Computing) 2013, ECOWS (European Conf. on Web Services) 2010, and Software Composition 2008. He also initiated the Workshop on RESTful Design (WS-REST) at the WWW conference. He's an advisory board member of EnterpriseWeb and a senior member of IEEE. Contact him at c.pautasso@ieee.org, and follow him @pautasso.

OLAF ZIMMERMANN is a professor of software architecture and an institute partner at the University of Applied Sciences of Eastern Switzerland in Rapperswil, Switzerland. Previously, he spent 20 years in industrial research and development and in professional services. He's particularly interested in architectural decision making and design knowledge sharing. As a senior certified IT architect, he has contributed to many company-internal knowledge management initiatives (both successful ones and less successful ones). He's the author of practitioner articles and scientific papers (including award-winning ones), and a contributor to textbooks. Zimmerman received a PhD in computer science from the University of Stuttgart. He has been on the organizing committees of the OOPSLA (Object-Oriented Programming, Systems, Languages, and Applications) conference, the SATURN (Software Engineering Institute Architecture Technology User Network) conference, and WICSA (Working IEEE/IFIP Conf. on Software Architecture) and has been on the *IEEE Software* advisory board since 2011. Contact him at ozimmerm@hsr.ch or www.ozimmer.de.

References

1. D. O'Leary, "The Most Cited *IEEE Software* Articles," *IEEE Software*, vol. 26, no. 1, 2009, pp. 12–14.
2. L. Rising, "Why Can't We All Play Nice?," *IEEE Software*, vol. 29, no. 5, 2012, pp. 7–10.
3. H. Wesenberg and E. Landre, "Making Architecture Matter," *IEEE Software*, vol. 29, no. 3, 2012, pp. 21–23.
4. E. Richardson, "What an Agile Architect Can Learn from a Hurricane Meteorologist," *IEEE Software*, vol. 28, no. 6, 2011, pp. 9–12.
5. W.A. Risi, "Next-Generation Architects for a Harsh Business World," *IEEE Software*, vol. 29, no. 2, 2012, pp. 9–12.
6. J. Crabb, "The BestBuy.com Cloud Architecture," *IEEE Software*, vol. 31, no. 2, 2014, pp. 91–96.