XP 25th Anniversary Workshop & Panel Report: Innovating Software Solutions – Past, Present, and Future

Abstract

Software practitioners have adopted many new ways of working over the past 25 years. Change has been driven by a diverse and global community of users, practitioners, researchers, and vernacular programmers. What have we learned over the past 25 years? What skills will software researchers and practitioners need in the future? Will AI or other emerging technologies offer opportunities for greater achievements, or will they become an obstacle to the human touch needed to develop software products? This paper reports on a combined workshop and panel organized and facilitated by Steven Fraser (Innoxec) together with Dennis Mancl (MSWX Software Experts) and Werner Wild (Evolution Consulting). The workshop and panel were part of the 25th Anniversary Track at the XP 2024 conference held in Bolzano, Italy.

Part I: Workshop discussion

More than a dozen participants from Europe and the Americas participated in the workshop. Discussion was structured around software innovation successes, challenges, and the future.

Successes

The workshop attendees observed improvements in software innovation over the past 25 years. The improvements included the core elements of Agile development practices: automated testing, rapid feedback, accelerated development and delivery at a sustainable pace, plus continuous improvement. Software frameworks and libraries have also made an enormous impact on quality and productivity. The adoption of frameworks combined with software libraries has reduced what Fred Brooks [1] referred to as “accidental complexity.” Peopleware [2] issues are now better understood and mitigated – inspiring collaboration through tools and techniques such as pair programming plus new ways of integrating teamwork and customer partnerships into software innovation.

During the workshop discussion, it was noted that the number of *vernacular programmers* [3] is increasing. These are practical computer users who are not trained/educated as programmers but who craft software to achieve scientific, business, or personal goals. For these developers, “design and coding” is the manipulation of spreadsheets, databases, web authoring systems, or other tools.

Challenges

The workshop participants brainstormed various challenges associated with past and present software practices. Workshop participants suggested that team members may feel a sense of being overburdened with “busy work” – e.g., meetings or too many Slack channels. Team turnover also creates gaps in “organizational memory.” There is a loss of know-how, and companies that rely on “gig economy” models are possibly the worst offenders since staff come and go frequently. In the past, innovation was catalyzed in corporate labs, but today’s product

innovations are more likely to be sourced from startups. Workshop participants suggested that new communications strategies might help inspire and better spread innovation.

Poor communication creates extra burdens. Teams are confused and can't meet business goals because of a lack of business awareness. Teams suffer from poor organization and communication gaps due to geography and/or developer-customer relations. Some teams struggle with culture conflicts, while others find that poor collaboration reduces the amount of learning across the team.

"Agile fatigue" seemed to be a common issue. Fatigue may become evident if teams are constantly pushed to adopt the latest tools and practices. Another discussion focused on the extra effort required to manage project iterations when there is a complex and evolving product backlog. Libraries and frameworks help to accelerate development, but when they evolve frequently, rapid change may spawn more disconnects and knowledge gaps.

"Fake Agile" was identified as another challenge. Workshop attendees suggested that some managers may mandate long lists of required tools and practices, even when they are unneeded. However, to be effective, agile development practices need to be customized somewhat for each new project "context." For example, when Scrum Masters and Product Owners are tasked with project management responsibilities, it may reduce their ability to keep their team's development processes lightweight and flexible.

Plan-driven versus iterative

"Working with business executives" is another challenge faced by development teams. Some leaders choose to give teams more independence, while others prefer to enforce a hierarchical management structure to guide planning and execution. The workshop participants discussed a troubling trend – development organizations that oscillate between "iterative" and "plan-driven" approaches. It is an understandable dynamic, because each approach has its benefits. Organizations should not be limited to a binary choice of "pure" Agile versus Waterfall development strategies: instead, teams should focus on practices appropriate to specific contexts.

Addressing today's software product innovation challenges

The workshop participants offered observations on future directions for software innovation, but they reached no firm conclusions. The adoption of emergent technologies such as AI or hardware acceleration will likely influence how development practices roles will change as technology evolves. These technologies may reduce the amount of direct human engagement in software design. On the other hand, they may increase the need for humans to focus on testing and configuration management.

To meet today's challenges, software development teams should improve their skills in communication and collaboration: partnering both internally (within their organization) and externally (customers, open-source communities). In today's world, it is important for teams to work with global, virtual, and external open communities. It was also observed that

development practices can be truly innovative when they focus on outcomes, not just on the approach or process.

However, collaboration challenges will likely not be solved by the wave of new artificial intelligence tools. AI may miraculously "shrink the work" of developers, but collaboration is a human endeavor.

Dealing with churn and issues of predictability will continue to be a challenge – be it from unstable software libraries, stakeholder interactions, delivery processes, or evolving software practices. In many companies, there is a natural tension between Operations and Development. Ops prefers a stable configuration, and may feel that Dev teams release too frequently. Organizational maturity can reduce churn and improve predictability; education and coaching are also factors. The goal is to build a better business – to enable business stability and growth.

Part II: Panel discussion

Panelists included: Alberto Brandolini (an Italy-based IT consultant, the author of *Introducing EventStorming* and the founder of the consulting firm Avanscoperta); Brian Fitzgerald (a software researcher and Principal Investigator at Lero, the Irish Software Research Centre, and a professor at the University of Limerick, where he also served as Vice President Research); Marko Hirsimäki (an Agile Coach at RELEX based in Helsinki, Finland specializing in Agile development and designing applications and architectures), and Diana Larsen (a US based Leadership Agility Advisor and author of *Lead without Blame*; *Agile Retrospectives*; *Liftoff*; and *Five Rules for Accelerated Learning*).

The panel began with a question posed by the workshop: What might follow – ad-hoc, Waterfall, and Agile software development models "post-Agile"?

Brian and Diana observed that Agile software practices emerged as a rebellion against the notion that there was "a single right way to develop software." For most of us in software development roles, software requirements are in flux. Generally, it isn't practical to follow a "rational process."

In his XP2024 keynote, Brian had reinforced this point by referencing the 1985 paper "A Rational Design Process: How and Why to Fake It" [4]. Parnas and Clements explained that developers aspire to design in a rational way: "Most of us like to think of ourselves as rational professionals. However, to many observers, the usual process of designing software appears quite irrational. Programmers often appear to make decisions without having reasons." Their conclusion: "The picture of the software designer deriving his design in a rational, error-free, way from a statement of requirements is quite unrealistic." Projects and teams face process challenges. The paper described the most common knowledge gaps: incomplete and changing requirements, lack of domain experience, complexity, design mistakes, and errors in internal documentation. The authors believed that an "ideal process" is impossible, so they suggested strategies for "faking it."

Diana asserted that we need to constantly update our methods. "The world is changing fast and we need to become learners. We can't just rely on old best practices, old knowledge. We need to be refreshing that all the time."

Marko and Alberto agreed. Marko stated: “My goal is to find better ways to work all the time.” Alberto noted that he would frequently change his approach, or he would apply multiple approaches to the same problem. Even though he has a recipe, he tries to solve problems with multiple approaches. “It doesn’t work this way, so let’s try this other way, and then try this way.” But not everyone is so flexible. Alberto complained that some processes aren’t staying flexible: “I’m seeing an inertia towards rigidity.” He also lamented that Agile suffers from “bad marketing.”

The context of software developers today looks a lot different from 25 years ago. Software intensive companies are more focused on cost-cutting and delivering value. Programming languages and tools have evolved considerably since the turn of the century and software practitioners are struggling to keep pace with advances in cloud technology and AI.

Unfortunately, many trainers and coaches reference old and dated principles, practices, tools, and publications that are no longer applicable. Brian called for more flexibility: “All of those [Agile] principles have to be taken in context.” Development teams need to use their judgment to decide which parts of their work might require “documents” (for example, to communicate component interface details with third parties). In some cases, a plan-driven approach can work effectively and efficiently because the system’s feature set is already known and documented.

Diana explained that many ideas that were incorporated in “Agile” existed for many years prior to 2001 [5]. For example, “customer satisfaction” was part of Total Quality Management of the 1980s. Customer issues will remain important in the future. Diana wanted to know “what will endure” – what are the parts of Agile that will persist beyond the marketing name.

How to talk about Agile principles and roles today

The discussion veered in the direction of “how to address the generation gap” between seasoned experts and today’s youthful practitioners. One of the audience members, JB Rainsberger, an experienced software consultant from Canada, offered this point of view: “The things that last are either the things we take for granted, or they are the things that we are constantly rediscovering.” If we rediscover an idea, it might satisfy a current project need. But shouldn’t we have read about this in the literature or learned it in a course?

Learning is hard. We are constantly rediscovering useful ideas and approaches for software development – “because they are important.” If young folks are struggling to learn these principles today, we shouldn’t be surprised. These ideas were just as difficult 25 years ago for the pioneers of the Agile movement.

JB thought that coaches need to be more positive and supportive. When speaking to today’s software teams, rather than “preaching” we should be more engaging. We could say, “We wanted to go back to the arms of Waterfall. We tried it, but it didn’t work. You’re going to try it, and it probably is not going to work for you either. Welcome to the club! Let’s figure out what we can do about that together.”

Panel comments suggested that it is time for the younger generation to drive a new software revolution. The next generation should craft the next set of software development principles

and practices. It's a mistake to mindlessly adopt dated practices from 25 years ago. Software developers face new development tasks and they are working in new contexts. The panelists explained how new development practices might require new roles, new responsibilities, and different job titles. Role titles can be useful to raise visibility for emerging technologies.

Brian suggested that AI is a new context. "You will have new ceremonies, roles, and artifacts to bring AI into the agile world." Brian said that one obvious role is "training the AI language model." Alberto explained that for today's generative AI systems, "prompt engineer" is an emergent role. (A prompt engineer [6] creates and optimizes the inputs to generative AI systems, possibly with the help of other AI systems.) Alberto suggested that we may also need to remove AI-generated garbage, so "content cleaner" might be another new AI-related role. A content cleaner would clean up the content and code generated by AI.

Education

Steve (Panel Impresario) asked the audience for their experiences in software education programs to answer the question "how you are educating your students on how to be more innovative," and how to look beyond just coding.

Peggy Gregory from the University of Glasgow made several observations. She explained "Students are still grappling with learning a lot of new technology," and there is considerably more technology than when she was a student. Teaching the development process isn't easy. Glasgow's curriculum requires that most software students participate in a third-year project – working on a team and building systems for real customers. But "learning Agile" is a challenge. "We put them in teams and we give them an Agile coach. But they still behave like students."

"Even after a four-year degree, their real interest is in learning the technology [languages, tools, AI]. They are much less interested in learning all of the other things – things that will become more important to them after they leave, but we can't yet persuade them how important they are."

AI and software development

There were a few comments about AI's potential contributions to software innovation. For example, will AI replace developers?

Brian observed that AI hasn't made it to the undergraduate curriculum yet at the University of Limerick. AI education is mostly directed at graduate students at present. Because we aren't educating average programmers and software engineers in AI, he sees the technology as "a long way from being available." While Brian could see AI as a decision support tool, he wasn't impressed by the software generated by AI.

Diana was concerned that AI might take over her writing: "I want to keep the fun part and have AI do the drudgery part."

Panel wrap-up

Each of the panelists had their own spin on software innovation.

Marko was a forceful advocate for innovation: “I don't want to go back. We need to keep reinventing.”

Brian emphasized the need for diversity. We need to expose young people to technology, the experience needs to be fun, with learning from role models.

Diana suggested that new ideas (which may define new roles and career opportunities) will always be emergent. Diana shared own experience in the early 1980s. She decided to connect to the technical world, but not everyone needs to be a coder. Deciding “what software needs to be written” is just as important.

Alberto shared his thoughts about the balance between profitability and sustainability. Alberto viewed profitability as the top goal for many companies. A project might face a choice when building some new features – whether to make the design of that feature more sustainable while reducing the potential profitability. Maybe within the constraint of profitability, we will try to be as sustainable as we can.

Summary: Innovating Software Solutions

Software innovation has evolved greatly over the past 75 years. Advances in technology have improved hardware capabilities in speed, capacity, and connectivity. In parallel, software processes evolved from plan-driven (Waterfall) to those that are more iterative (Agile). Future software innovation must go beyond today’s AI and cloud computing frameworks. We must discover new ways for individuals and companies to innovate.

Both the workshop and panel observed that the context for software innovation has changed over the past 25 years. As technologies have evolved, peopleware issues are better understood, and companies are more sensitive to the quarterly drumbeat of the stock market. There was consensus that it is wrong to mindlessly apply dated principles without regard for new contexts and technologies.

“Agile fatigue” and “Fake Agile” are symptoms of a context mismatch between today’s reality and some Agile principles based on a “25-year-old Manifesto.” In his XP 2021 Keynote talk [7], Steve McConnell argued for updating “Agile” with principles such as continuous testing and limiting work-in-progress. Workshop attendees suggested that “busy work” was becoming more of a problem – and we should look forward to a younger generation of developers and researchers leading a new software technology revolution.

Today’s software innovation depends on increased “business awareness” by development teams. Without business awareness, team and company longevity is at risk. Software is now a core part of every system, and more systems have customer-visible requirements, such as security, safety, and responsiveness.

University curriculums are evolving in response to both student interest and company talent needs. Our universities need to be prepared to teach both new technologies (AI, data science tools, vernacular programmer tools) and new post-Agile software development processes. Global innovation accelerates when relevant software practices can be taught and learnt with less effort.

While there were no absolute predictions for the future, both the workshop and panel observed that people issues are critical to software innovation, possibly more important than AI.

References

1. Brooks, F.P.: No Silver Bullet – Essence and Accident in Software Engineering. Computer, 20 (1987), no. 4, 10-19. https://doi.org/10.1109/MC.1987.1663532

2. DeMarco, T., Lister, T.: Peopleware: Productive Projects and Teams (3rd ed.). Addison Wesley, Boston MA (2016).

3. Shaw, M.: Myths and misconceptions: what does it mean to be a programming language, anyhow? Proceedings of the ACM on Programming Languages, 4 (2020), Issue HOPL, 234–277. https://doi.org/10.1145/3480947

4. Parnas, D.L., Clements, P.C.: A rational design process: How and why to fake it. In: Ehrig, H., Floyd, C., Nivat, M., Thatcher, J. (eds) Formal Methods and Software Development. TAPSOFT 1985. Lecture Notes in Computer Science, vol 186. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-15199-0_6

5. Larman, C., Basili, V.R.: Iterative and Incremental Development: A Brief History. Computer, 36 (2003), no. 6, 47-56. https://doi.org/10.1109/MC.2003.1204375

6. Genkina, D.: Don't Start a Career as an AI Prompt Engineer. IEEE Spectrum, 61 (2024), no. 5, 30-34. https://doi.org/10.1109/MSPEC.2024.10523015

7. McConnell, S.: 20 Years is Enough! It's Time to Update the Agile Principles and Values. XP 2021 Conference, https://www.agilealliance.org/resources/sessions/20-years-is-enough-its-time-to-update-the-agile-principles-and-values/. Accessed 21 Jul. 2024

[This report will appear in the proceedings of the XP 2024 Conference, published by Springer Nature.]