No Silver Bullet Reloaded: Report on XP 2017 Panel Session

Steven Fraser
Innoxec
Santa Clara, CA, USA
sdfraser@acm.org

Dennis Mancl
MSWX Software Experts
Bridgewater, NJ, USA
dmancl@acm.org

ABSTRACT

At XP2017 in Köln, a panel was convened to discuss the classic 1987 IEEE Software paper by Frederick P. Brooks, "No Silver Bullet: Essence and Accidents in Software Engineering." The ideas presented in his paper have influenced several generations of software developers. Brooks emphasized the notions of essential complexity and accidental complexity, and he offered suggestions for promising approaches to software development. While his approaches are linked to what we now recognize as "agile practices," panelists offered an implicit caveat that they must be done with discipline to avoid increased accidental complexity. Panelists also observed that agile development itself is not a "silver bullet".

Categories and Subject Descriptors

D.2.9 **[Software Engineering]**: Management; K.4.3 **[Computers and Society]**: Organizational Impacts; K.6.3 **[Management of Computing and Information Systems]**: Software Management

General Terms

Management, Design, Economics.

Keywords

Silver bullet, complexity

1. NO SILVER BULLET AND AGILE DEVELOPMENT

Frederick P. Brooks described the issue of "essential versus accidental complexity" in his classic 1986 paper [1-2]. "Essential" is complexity related to the conceptual structure of the system to be developed, "accidental" is other complexity usually coming from the development process. He claimed that to make an order of magnitude improvement in software development, it is necessary to attack essential complexity, unless accidental complexity is responsible for more than 90% of the development cost.

Brooks proposed some "promising approaches" (buy versus build, requirements refinement and rapid prototyping, incremental development, and supporting great designers). Agile development might help to reduce both essential and accidental complexity. The panelists promised to take on the question of what we need to do if we want agile development to be effective.

Panelist positions focused on two issues:

- Complexity may increase faster than our ability to deal with it. How can teams keep up with the demand for complex software?
- There are challenges in training and mentoring a new generation of software developers. How can developers prepare within the traditional college and university education programs?

In the course of the panel discussion, the panelists and audience explored the following ideas that are important for all agile organizations today.

- Agility is a good approach to reduce essential complexity, even though it is not a "silver bullet." Agile teams need to work to keep their designs simple, and they use iteration and automated testing to get internal feedback to control the growth of complexity.
- Globalization creates more accidental complexity, because development teams spread across multiple locations and countries have more difficulty coordinating their work.
- Agile team members need to master a set of technical practices, but it is also important to keep a focus on the needs of the business. In the popular view, agility is merely a set of techniques for technical staff, but practitioners believe that "business agility" is critical for building software products.
- Brooks said it is important to "develop great designers." We need to recognize that design skills go beyond just software craftsmanship. Agile development emphasizes teamwork and communication, so great designers must have good people skills.
- Good designers learn from failure. Agile teams follow an "experimental way of working" to deal with complex change. Documenting failures can help others learn from mistakes.
- It is important to think about the education of the next generation of software designers. Unfortunately, many college and university programs only offer limited experience in industrial-scale software development and teamwork. This might be improved with increased collaboration between industry and academia.

2. OPENING STATEMENTS

Steve Fraser (Innoxec, USA) explained the main propositions of Fred Brooks' "No Silver Bullet" paper: Software development organizations can improve in their management of both essential complexity and accidental complexity by improving the training of software professionals, using more iteration, and using tools that improve the ability of developers to visualize their software. Steve also indicated that there are new kinds of accidental complexity today – and the new-style accidental complexity is tied to modern development challenges such as global development, developers from different cultures, and increased vulnerability to cybersecurity risks.

Jutta Eckstein (Independent Consultant, Germany) has coached both large and small teams. She worried about the increase in complexity that we all face – both from large team sizes and from the fact that our systems have more complex requirements. Jutta pointed out that agile development is not a "silver bullet" in itself, but agile can help reduce complexity in several ways. Agile organizations work hard to keep their designs simple, and they use tools and technologies (unit test frameworks and continuous integration) that improve internal feedback. Jutta also reported that many organizations don't focus enough on managing complexity.

Andreas Schliep (Das ScrumTeam AG, Switzerland) works as a Scrum coach and trainer. He suggested that we need to measure and monitor complexity in our software. He also indicated that complexity can result from a gap between developers and testers, or a gap between managers, developers, and product people.

Hendrik Esser (Ericsson, Sweden) works for a large global telecom company, and large companies have special problems – dealing with globalization, disruptive technology, and continual internal reorganizations. He observed that agile development has been

responsible for big improvements in productivity, but that globalization is creating more accidental complexity – due to the challenges coordinating the work across locations and cultures.

Ademar Aguiar (Universidade do Porto, Portugal) explained how software development is a field where knowledge management and knowledge transformation is critical. As an academic, Ademar has been interested in how to capture more of the tacit knowledge that is in the minds of people, using simple tools such as wikis to improve team communication.

Werner Wild (Evolution Consulting, Austria) explained that artificial intelligence won't solve all software development problems. He was worried that both accidental and essential complexity are increasing much faster than we can adopt new practices and new paradigms of software development. Werner pointed out that the business world has become more focused on business agility, but he felt that this trend might detract from a focus on the more technical aspects of software product development.

3. QUESTIONS AND DISCUSSION

3.1 Attention to Technical Practices

Do teams have enough skill and knowledge technical parts of agile? The discussion was initiated with and audience comment: "I worry about companies that hire agile coaches who don't understand the technical practices we need to create high quality software." The point was that some agile coaches have moved teams towards more management-focused agile practices from the Scrum world.

Andreas agreed that current "Scrum Master" training was too superficial and observed that Scrum Masters can make their living without coding knowledge. However, to be effective they must be aware of – and promote the use of – engineering best practices

Andreas also asserted, "There is no such thing as 'business agility'. The term is stupid. Agility is about being able to fulfill business demands by applying appropriate engineering practices." Andreas continued to emphasize that agile is about engineering. "There is a misconception that the business view and the technical view of the problem are not related. You can only achieve 'business agility' with a proper set of engineering practices – with a proper toolbox."

Jutta immediately disagreed and suggested that "company-wide agility" might be a better designation than "business agility." Jutta suggested that companies need to be flexible, adaptable, learning and able to focus on the customer – not just try to follow the Agile Manifesto. An important part of agility is running experiments, which help teams learn more and move forward.

3.2 Do Educators Have Industry Experience?

The panel agreed that university students are learning from academics who are not necessarily "practitioners" – what can we do to ensure that the people who teach agile are connected to real-world experience?

Both Andreas and Steve suggested that the educational system should incorporate real-world experience well before the students reach "higher" education. It should begin in preschool. And it is a challenge to change the education system to get more real-world practices into the curriculum.

Steve mentioned that Oracle is developing a "Design Tech" high school [5] close to its headquarters in Redwood City, California. "It's going to be an IT high school. They want to teach programming skills and the skills we don't often see until you get a real job."

Andreas thinks that education is important for achieving organization maturity. "I partly agree with Jutta that the lack of encompassing the whole organizational agility is something that not only fosters accidental complexity, I think that's one of the main reasons that there is organizational conflict – the head doesn't align with the body."

Steve called for more academia-industry collaboration. "Many companies will try to have internship programs... like the apprenticeship programs that we see in other more trade-like vocations. It's through those internships or fellowships where you get practical experience. Another thing is sabbatical programs, where professors are brought to industry."

Jutta thinks that schools need to help students understand different kinds of "change." She described the three categories of change from Human Systems Dynamics (HSD) – simple change, dynamic change, and complex change [3]. Simple change is when you are moving something between two known states and the pathway is straightforward. Dynamic change is where the pathway is not clear, so you move in a series of small steps – maybe with a set of milestones to check. Complex change is where the destination is unclear, so the pathway is a series of experiments to see what improves the current state. There are a lot of people in corporations who don't understand complexity and change. Agile development is a valuable approach to deal with dynamic change. Jutta believes that software developers must learn to be more experimental in their work.

3.3 Brooks and Development of Great Designers

How do we develop great designers? University education isn't always effective: they don't get enough feedback because we don't know what to measure. Design is very hard to learn – generally people learn more by experience in the company of others than in instructor-led learning.

Great designers are not easy to find. Hendrik explained one of the staffing problems faced by corporate managers. "Productivity is important in business. We want great designers because they can produce things faster and we can get our return on investment faster. We just don't get enough of these people from the universities."

Hendrik complained that the merit systems in our companies don't give a person the incentive to become a superstar designer. There is more of a push to be a superstar manager. He also pointed out that cross-functional teams are a good practice to increase learning on the job.

Ademar noted that design is difficult to teach in the university. "We give students many examples of things so they can imitate first, and after a while they can craft their own. The most effective way is to give examples, real examples from industry, with real customers, so they know what they are doing."

Andreas bemoaned the lack of time to learn. "We know that the university education and the overall education is insufficient, and we need to give teams and people room to learn. But this doesn't happen. I worked with a team last week on engineering practices, and they asked the question 'is it OK if we take one or two story points in order to address the most urgent technical debt issues?' They are being pushed too hard – they just don't have time to address the stuff that is affecting their system, so there is no opportunity to learn."

Hendrik and Werner shared the observation about the importance of working together and learning from others.

Hendrik explained that human skills can be a problem. "All the accidental complexity that we are confronted with may be a sign that interaction between people and coordination skills are not so good – this is a skill we are really struggling to build with the teams. So instead of just software craftsmanship, we need to address human skills."

Werner explained that we need to learn from past mistakes. He advocated for creating a collection of software failures, similar to the public database of failures created from the analysis of aviation accidents. "In software it is very hard to learn from the failures of others, except if you happen to know someone and go for a beer and you get the real story."

Steve pointed to one comment made by Brooks at a panel in 2007: "I know of no other field where people do less study of other people's work." [4]

3.4 How Can Students Get Experience?

The panel turned to the question of practical experience for students. Students never get to experience a large software development project.

The panelists all agreed that students would learn something from working in a team of 40 people.

Werner started by promoting the idea of more long-term work in school. "Run one-year labs and have the next group of students maintain and extend the system. This is one thing that most students are not exposed to – how to maintain a system, how to evolve a system over time."

Ademar pointed out that at University of Porto, students get to work as part of a large team in a fourth-year software course – working on a project with real companies. The students take two courses. In the first semester, they work in teams of six students and they work with a not-for-profit organization with a social cause or social venture. In the second semester, they work on projects with real companies, and they get experience working in a team of 40 people.

Ivana Gancheva (SINTEF Digital, Norway) warned that the job of a university is not to just match today's needs of industry. Students should bring new technology to industry. "We don't want to just duplicate the mess in industry – we want students that can maybe throw away some old stuff."

Werner explained that he works as a university instructor and as a consultant in industry, and he brings in some of the best students to work on industry projects. It is easier for students to get started with smaller industry engagements – not so easy to have them working with hundreds of people in a big company.

Hendrik also explained about some efforts in education to teach Scrum-based problem-solving techniques. He mentioned a Dutch education website called eduScrum: http://eduscrum.nl/en – which has some interesting ideas that some people might wish to try.

4. SUMMARY

The panelists explored the impact of agile methods on a wide range of software challenges: dealing with complexity, training and mentoring, focus on business needs, and learning from failures. The ideas of essential and accidental complexity are still relevant if we want to do a better job of applying agile methods. The "promising approaches" to complexity that were outlined by Brooks 30 years ago are still valuable in today's more agile world: reuse, prototyping, incremental development, and supporting great designers. The panel concluded with observations that the agile community can benefit by revisiting some of the classic software engineering ideas.

5. REFERENCES

[1] Brooks, F.P. 1986. No Silver Bullet: Essence and Accidents of Software Engineering. In: *Proceedings of the IFIP Tenth World Computing Conference*, 1069–1076.

[2] Brooks, F.P. 1987. No Silver Bullet: Essence and Accidents of Software Engineering. *IEEE Computer*, 20, 4 (Apr. 1987), 10-19.

[3] Eoyang, G.H. and Holladay, R.J. 2013. *Adaptive Action.* Stanford University Press, Stanford, California.

[4] Fraser, S., Mancl, D. 2008. No Silver Bullet: Software Engineering Reloaded. *IEEE Computer*, 25, 1 (Jan. 2008), 91-94. DOI= http://dx.doi.org/10.1109/MS.2008.14.

[5] Website of Design Tech High School. 2018. http://www.designtechhighschool.org. Accessed 31 January 2018.