



# Toward Learning Teams

Rashina Hoda, Jeffry Babb, and Jacob Nørbjerg

**TODAY'S FAST-PACED WORLD** of software development is filled with uncertainties that demand agility. With rapid changes in technology and platform terrains, software teams are more than ever expected to hit the ground running—even as they frequently find themselves in uncharted territories. One strategy for surviving this onslaught of ever-changing conditions is to fight variety with variety.<sup>1</sup> In other words, equip development teams with a variety of skills and abilities to effectively handle the variety of changing technologies, platforms, and requirements they face on a daily basis.

To expect a team to have every skill upfront is to expect a perfect team. Even if this idealistic vision were achievable, the team would still need to frequently upgrade its skills to keep pace with ongoing changes. So, perfect teams aren't the answer to today's software development challenges. Rather, we need *learning teams*—teams that can repeatedly bend and blend on demand to suit the environments they work in.

Ideally, software teams are meant to engage regularly in practices that enhance various types of learning—in new engineering and management practices, new or complex technical skills, cross-functional skills, and experiential lessons learned. All these knowledge areas fuel continuous improvement, a core principle of agile methods: “At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly” (<http://agilemanifesto.org/principles.html>).

Here, we share evidence from longitu-

dinal qualitative studies based on observations and queries of practices in several agile teams. Although the studies didn't focus exclusively on learning issues, they revealed the challenges of keeping continuous learning practices at the same level of importance as delivering working software.

The biggest obstacle to such learning is *iteration pressure*—that is, the pressure experienced by teams to deliver goals committed to within an iteration. Although some iteration pressure helps motivate teams to meet their goals, working in high-paced, uncertain environments and catering to several project commitments at once often leads to sacrificing core agile principles with regard to continuous learning.

## Evidence of Iteration Pressure

Our article combines and compares findings from two studies:

- a grounded-theory study of software teams and practice in medium to large organizations involved 58 agile practitioners across 23 software organizations in New Zealand and India—most teams were using Scrum, XP, or a combination of the two;<sup>2</sup> and
- a longitudinal action-research study of reflective practice in a small US software development company using agile practices.<sup>3</sup>

The qualitative nature of the research design allowed us to observe and query team members directly. In both studies, the teams



adapted standard agile practices to suit particular organizational, project, or domain contexts.<sup>4</sup> However, the practices most closely related to learning were also those most often sacrificed or poorly implemented over time. The paradox is that continuous learning is essential to maintaining a team that can continue its effectiveness.

Time constraint was commonly cited as a reason for learning-related problems. For example, some teams faced significant—and at times excessive—iteration pressure on a regular basis:

*I'm always feeling the need to rush, rush, rush.... After one week [iteration], we want to remove all these stickies [task cards] from the wall. So it's always pressure. —Developer, New Zealand*

Software practitioners expressed their concerns about “fitting learning” into an iteration. They felt that customers pay for development of features—

*So focus is on delivering business value as soon as possible. As a result of that, you take items which are most required from the point of view of business. —Developer, India*

Adaptations made in response to such time constraints are usually unintended and unplanned, so their consequences are difficult to handle. For example, as a consequence of iteration pressure, teams struggle to set aside time from their regular development tasks for learning new techniques and skills:

*I'd be interested to learn various agile techniques for requirements gathering..., [but] I haven't really had a lot of time to think about it. [Scrum] is very action oriented. —Business analyst, New Zealand*

As teams start to feel iteration pressure, specific learning-focused practices, such as retrospectives,<sup>3</sup> start to drop down their priority list.

business, and it's somewhat common to be living from project to project. As a smaller team works toward growing its business, the exposure to iteration pressure is very direct because there's no management layer above the team members and they're directly exposed to customers.

*There's no way that I can keep up with the technology and grow the business at the same time.... [I don't] have the time to stop, reflect, and learn. —Small software organization owner and lead developer, US*

Additionally, teams must find time just to keep up with advances in new technology. In the face of time scarcity, where can spare cycles for reflective team learning be found?

*[For a] small group to be able to make time for learning, [you realize that] it's the prudent thing to do, but you would have to really force yourself to do so. —Small organization developer, US*

The practices most closely related to learning were also those most often sacrificed over time.

tangible and testable—written as user stories, estimated by story points, and implemented as technical tasks. Although some customers—typically, long-term with established trust—appreciate the value of continuous learning, others are reluctant to pay for activities that aren't directly linked to immediate feature development. The benefit of including something as intangible as learning into iterations is difficult to justify.

*In the [retrospectives] that we do, they are so much quicker now than it used to be.... And now it is almost like lip service.... We don't do self-evaluation as well as we used to. —Tester, New Zealand.*

Some adaptations are made in the context of very small development companies akin to start-ups, where the team is the organization. These teams face the pressure of procuring enough

## Learning under Pressure

Several strategies emerged from our studies—both preventive and curative—for achieving a balance between iteration pressure and continuous learning.

## Expectation Management

Managing expectations is a preventive strategy to contain iteration pressure. In the case of new agile teams, making room for initial learning involves educating customers:

*I have sort of a secret conversation with the customer, “Right, okay. This team is new here for learning. Expect them to blow the first sprint. It is very likely to happen.”... And if anything good comes out of it, they [custom-*

ers] are positively surprised. —Agile coach, New Zealand

Similarly, you can manage the expectations for mature teams by educating management and customers on the importance of learning. This leads to a learning team environment that lets teams learn from mistakes and have ample opportunities to include various types of learning within their regular iterations.

### Reflective Practice

This epistemological perspective holds that tacit knowledge is grounded in knowing-in-action and knowing-in-practice. That is, new knowledge and knowing are grounded in the outcomes of daily practice in which practitioners assess the outcomes of their daily experimentation to build their own repertoire of expertise.

This is both a curative and preventive strategy. It's curative in that teams can be refocused on creating the "tiny habits,"<sup>5</sup> which can serve to reestablish the primacy of learning. It's preventive because teams that focus on reflective practice will accept the value of even a small amount of time devoted to learning, seeing it as normative behavior that's as important as unit or acceptance testing.

### Retrospective

A retrospective is an agile practice specifically devoted to reflective practice.<sup>6</sup> It lets teams inspect their current state, learn from experiences, and make plans to adjust future iterations on the basis of that learning. Introducing and adhering to meaningful retrospectives is a key to sustained learning.

*The key here that makes it all work is this practice of retrospectives. Because that essentially says..., "Stop. How are we doing, guys?" ...[W]ith this practice and with the continuous kneading*

*out the things that don't quite work and focusing on the things that work, you grow that ecosystem, you develop it, and you're bound to be successful.*  
—Agile coach, New Zealand

### Learning Spike

A learning spike is an adapted, curative practice of setting aside exclusive time for learning, either within an iteration or spread across multiple iterations. It might not involve the whole team. While some members perform

balancing act between iteration pressure and continuous learning. If your balance falls on the side of iteration pressure, you lose out on learning. If your balance falls on the side of continuous learning, you lower productivity in the short term. Ideally, a state of perfect balance or equilibrium between the two ensures good productivity with learning.

Small but frequent investments in learning would likely avoid mammoth one-time repayment scenarios that can bring team productivity to a complete

Reflective practice grounds new knowledge and knowing in the outcomes of daily practice.

the learning spike, others can continue to work on regular stories and tasks, thereby managing iteration pressure to an extent.

For example, practitioners will often attend technical conferences, go for more training in a workshop, or perhaps even undertake an internal retreat.

### The Inevitable Payoff

The dilemma faced by software teams is between harnessing opportunities for continuous learning on the one hand and sacrificing or postponing learning-focused practices and succumbing to iteration pressure on the other. On an everyday basis, teams must choose between quick solutions for short-term gains and investments in good practices for long-term benefits. This is similar to the "technical debt" idea that Ward Cunningham introduced in 1992,<sup>7</sup> except it focuses on learning—hence, a *learning debt*.

Managing the dilemma requires a

standstill. For example, apart from regular practices such as retrospectives, every few iterations the team might decide to slow down and invest in other learning-focused practices, such as a learning spike.

### The Missing "I" in Teams?

Perhaps the missing element in many strategies for sustaining team learning is each team member's individual responsibility for engaging in constant and daily reflection practices. Because agile methods call for self-motivated and cross-functional teams, the various progenitors of agile methods likely envisioned effective agile team members as reflective practitioners.

Donald Schön's reflective-practitioner perspective suggests that the onus of learning under pressure is on the individual practitioner.<sup>8</sup> This isn't a panacea but a mindset to be infused into team culture. For instance, Schön suggests adopting a daily reflection and

# IEEE Software

## HOW TO REACH US

### WRITERS

For detailed information on submitting articles, write for our Editorial Guidelines ([software@computer.org](mailto:software@computer.org)) or access [www.computer.org/software/author.htm](http://www.computer.org/software/author.htm).

### LETTERS TO THE EDITOR

Send letters to

Editor, *IEEE Software*  
10662 Los Vaqueros Circle  
Los Alamitos, CA 90720  
[software@computer.org](mailto:software@computer.org)

Please provide an email address or daytime phone number with your letter.

### ON THE WEB

[www.computer.org/software](http://www.computer.org/software)

### SUBSCRIBE

[www.computer.org/software/subscribe](http://www.computer.org/software/subscribe)

### SUBSCRIPTION CHANGE OF ADDRESS

Send change-of-address requests for magazine subscriptions to [address.change@ieee.org](mailto:address.change@ieee.org). Be sure to specify *IEEE Software*.

### MEMBERSHIP CHANGE OF ADDRESS

Send change-of-address requests for IEEE and Computer Society membership to [member.services@ieee.org](mailto:member.services@ieee.org).

### MISSING OR DAMAGED COPIES

If you are missing an issue or you received a damaged copy, contact [help@computer.org](mailto:help@computer.org).

### REPRINTS OF ARTICLES


For price information or to order reprints, send email to [software@computer.org](mailto:software@computer.org) or fax +1 714 821 4010.

### REPRINT PERMISSION

To obtain permission to reprint an article, contact the Intellectual Property Rights Office at [copyrights@ieee.org](mailto:copyrights@ieee.org).

action cycle called the “ladder of reflection,” which James Tomayko and Orit Hazzan operationalize into a means of habitualizing daily reflection in and on action.<sup>9</sup> Such a simple practice presents a “tiny habit” toward incremental and iterative learning.<sup>5</sup>

**I**n some respects, agile teams become victims of their success because managers and customers come to expect a steady pace of productivity. When facing excessive iteration pressure, the teams we observed often succumbed to making decisions that favored quick solutions that would meet tangible iteration deadlines over investing in continuous learning to gain distant long-term benefits.

Effective learning under pressure involves conscious efforts to implement strategies that will balance its priority within iteration pressures. Teams, their management, and customers must all recognize the importance of creating learning teams as the key to braving the erratic climates and uncharted territories of future software development. 

### References

1. G. Morgan, *Images of Organization*, Sage Publications, 1986.
2. R. Hoda, J. Noble, and S. Marshall, “Self-Organizing Roles on Agile Software Development Teams,” *IEEE Trans. Software Eng.*, vol. 39, no. 3, 2013, pp. 422–444.
3. J.S. Babb and J. Nørbjerg, “A Model for Reflective Learning in Small Shop Agile Development,” *Proc. IRIS: Selected Papers of the Information System Research Seminar in Scandinavia*, Hanne Westh Nicolajsen, Trondheim, 2012, pp. 23–38.
4. R. Hoda et al., “Agility in Context,” *Proc. Object-Oriented Programming, Systems, Languages and Applications Conf. (OOPSLA 10)*, ACM, 2010, pp. 74–88.
5. B.J. Fogg, “A Behaviour Model for Persuasive Design,” *Proc. 4th Int’l Conf. Persuasive Technology (Persuasive 09)*, ACM, 2009; doi:10.1145/1541948.1541999.
6. E. Derby and D. Larsen, *Agile Retrospectives: Making Good Teams Great*, Pragmatic Bookshelf, 2006.
7. W. Cunningham, “The WyCash Portfolio Management System,” Addendum to *Proc. Object-Oriented Programming Systems, Languages, and Applications Conf. (OOPSLA 92)*, ACM Press, 1992, pp. 29–30.
8. D. Schön, *Educating the Reflective Practitioner*, Jossey-Bass, 1987.
9. O. Hazzan and J. Tomayko, *Human Aspects of Software Engineering*, Charles River Media, 2004.

**RASHINA HODA** leads the Software Engineering Processes, Tools and Applications research group in the University of Auckland’s Department of Electrical and Computer Engineering. Her research interests include self-organizing teams, agile software development, and human-computer interaction. Hoda received her PhD in computer science from Victoria University of Wellington, New Zealand. Contact her at [r.hoda@auckland.ac.nz](mailto:r.hoda@auckland.ac.nz).

**JEFFRY BABB** is an assistant professor of computer information systems at West Texas A&M University. His research interests include small-team software development, agile software development, and mobile application development. Babb received his PhD in information systems from Virginia Commonwealth University. Contact him at [jbabb@wtamu.edu](mailto:jbabb@wtamu.edu).

**JACOB NØRBJERG** is an associate professor in the Copenhagen Business School’s Department of IT Management. His research interests are in systems development organization and management and software process improvement. Nørbjerg received his PhD in information systems development from the University of Copenhagen. Contact him at [jno.itm@cbs.dk](mailto:jno.itm@cbs.dk).



Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.