Industrial and Organizational Psychology (2019), **12**, 272–276
doi:10.1017/iop.2019.85

CAMBRIDGE UNIVERSITY PRESS

COMMENTARY

Applying Lean to cognitively complex work

Bill Curtis*

CAST Software
*Corresponding author. Email: curtis@acm.org

Much of the writing and research on Lean has been conducted in business domains that are more procedural and routine, or to the more routine aspects of domains such as health care where nonroutine work is frequent. There is little literature on the behavioral issues raised when Lean practices are applied to work that is nonroutine and cognitively complex, such as software development. The challenges in this domain desperately beg for empirical research by industrial and organizational (I-O) psychologists, often in tandem with psychologists from other specialties such as cognitive science (Curtis, 1981; Curtis et al., 1986).

Software development is a product design discipline rather than a production or service discipline. The most complete explication of principles for applying Lean to software development was provided Mary and Tom Poppendieck (2003, 2007), who admit that Lean practices must be transformed to be effective in software. Their seven principles for Lean software development include the following: eliminate waste, build quality in, create knowledge, defer commitment, deliver fast, respect people, and optimize the whole.

In software development, Lean practices have been integrated with so-called "Agile" development methods, whose primary characteristics of small batch sizes and self-managed teams are hallmarks of Lean. Adoption of these Lean/Agile practices has grown this century in reaction to difficulties experienced in "Waterfall" projects where, in theory, the entire system is designed, then coded, then tested in a top-down progression. In contrast, Agile methods (a collection of development methods whose specific practices are no more clearly bounded than those of Lean) have adopted the Lean practice of breaking a project into smaller batch sizes (Reinertsen, 2009) of working software that can be delivered in short iterative cycles (typically called "sprints") of usually a month or less. This allows customer representatives who rarely know their full requirements at the outset to react to a working "batch" of software and refine their requirements as they experience the growing system, supporting another Lean principle of closer connection with customers. Some Lean/Agile projects adapt Kanban practices (Brechner, 2015) to control the queue of work, optimize work in progress, and ensure the development team avoids overload.

Like the Toyota Production System, Lean/Agile practices have transformed the way developers work. However, the widespread adoption of Lean/Agile has experienced difficulties (Meyer, 2014), some of which might have been mitigated by interventions from I-O psychologists. Three difficulties discussed here involve challenges with self-managed teams, culture clashes, and organizational learning. The nature of these challenges in cognitively complex work offers important opportunities for I-O research.

Self-managed software development teams

When Lean/Agile practices are adopted, teams are empowered to self-manage their own projects with a SCRUM master (SCRUM is the most popular Lean/Agile method) who is more a coach or

© Society for Industrial and Organizational Psychology 2019.

https://doi.org/10.1017/iop.2019.85 Published online by Cambridge University Press

mentor than a project manager. Self-management is believed critical to team efficiency and motivation during short development sprints. However, when Lean/Agile practices are adopted, teams are often empowered without any preparation or training in the responsibilities of self-management. Consequently, many executives are complaining about development teams that lack discipline and are not aligned with corporate objectives—well-understood problems in the literature on self-managed teams. Executives are also struggling with how to coordinate work among multiple self-managed teams on large projects.

The software development community is largely unaware of the substantial body of research results the I-O community has amassed on team building and empowerment. Guidance on creating team-based organizations (Mohrman, Cohen, & Mohrman, 1995) and high-tech teams (Bowers, Salas, & Jentsch, 2006) provides a foundation of evidence-based practices and preconditions for establishing effective self-managed teams. Software organizations would benefit from engaging I-O psychologists to guide the implementation of self-managed teams when adopting Lean/Agile practices.

In addition to guidance on team-building practices, there are numerous opportunities for team-based I-O psychology research in software development. For instance, some Lean/Agile Methods recommend allowing a system architecture to emerge as the system grows across multiple sprints rather than designing it at project initiation. What practices best help individuals on a team or multiple teams sustain a common mental model of a system architecture that is emerging and is typically not documented? How should progress and dependencies among multiple self-managed teams on large system projects be managed? What are the limits to empowerment and self-management when teams share multiple technical and schedule dependencies? How should individual and team performance be evaluated when the amount of code produced fails to capture the quality of its design or density of defects?

Culture clash

Another potential reason for the difficulty some organizations have experienced with self-managed software teams is the clash of an organization's culture and climate with the professional culture of software developers. Consider a "professional culture" to be a pattern of shared beliefs about values, practices, and ways of working among those in a specific domain of skill that they carry into the environment of their employer. There has been little attempt to measure the professional culture of software developers. One prominent attribute of this culture is its resistance to traditional instruments of accountability such as managers and measures. Developer culture relies more on trust, interpersonal interactions, and personal motivation. This professional culture clashes with organizational cultures employing traditional modes of accountability such as project managers, earned value analysis, productivity measures, and estimating a project's full cost up front.

Research is needed to identify characteristics of the professional culture of software developers and how they may be affected by or differ among generations, locales, education, organizational practices, and similar factors. I-O psychologists can help software organizations understand and reduce culture clash as a component of a change management program supporting the adoption of Lean/Agile methods. Research questions involve how to measure the amount of clash between professional and organizational cultures, and how it affects outcomes such as engagement, retention, and performance.

Team versus organizational learning

The professional culture of software developers focuses at the team rather than organizational level. This orientation, often described as tribal, can clash with organizational cultures that stress standardized practices, values, and results. Lean/Agile practices are adopted by teams to serve their

https://doi.org/10.1017/iop.2019.85 Published online by Cambridge University Press

objectives with less attention to organization-wide benefits. The team focus is especially noticeable in the deployment of knowledge created through continual improvement.

Many Lean/Agile projects conduct "retrospective" meetings at the end of each sprint (a Kaizen event) to identify lessons learned and opportunities for improvement that can be implemented in the next sprint. Retrospectives do not have a mechanism for communicating improvement ideas to other development teams in the organization. In addition, the measures used for estimating the amount of work to be undertaken in a sprint (called story points), and for evaluating progress toward completion, are calibrated uniquely to each team's ability to produce software. Consequently, one team's story points are generally not statistically comparable to the story points of other teams. Although retrospectives and story points benefit the team, they do little to deploy knowledge and improvement across the development organization.

To expand organizational learning, I-O psychologists could recommend practices used successfully in other team-based environments to create organizational learning from team experience and measures. I-O research questions could involve methods for overcoming resistance to standard measures and the most effective ways to spread complex technical knowledge about architectures, coding tricks, development tools, and root causes of defects across the organization. Research is also needed to demonstrate empirically how growth in the organizational spread of knowledge affects project outcomes and the speed with which developers become productive when transferred to other projects.

Why haven't I-O psychologists engaged Lean practices in cognitively complex work?

Balzer, Brodke, Kluse, and Zickar (2019) list three reasons that could dissuade I-O psychologists from conducting research or consulting in Lean. I will discuss each of these and then present a potentially larger challenge.

1. *Lean/Agile practices are a fad*. Twenty years ago, Lean/Agile practices could be considered a fad. However, their widespread adoption and replacement of older practices has moved them beyond the fad phase. Nevertheless, various practices are being reworked as weaknesses are identified. Given the growing footprint of software in business processes and products, guidance and research from I-O psychologists on Lean/Agile development methods and practices is critical and timely.
2. *Distinguishing Lean from other process improvement strategies is difficult*. Lean has drawn many practices from other process improvement techniques. Lean practices are so frequently integrated with practices from other improvement techniques that it is impossible, at least in software development, to distinguish Lean as separate from Agile methods and various approaches to continual improvement. Frankly, I-O psychologists should embrace the larger issue of research on continual improvement because there are few well-defined "pure" methods and their practices are frequently intermingled.
3. *The quality of research falls short of I-O standards*. Forty years ago, empirical research in software engineering fell far short of standards in I-O journals. However, empirical researchers with software engineering or related backgrounds have substantially improved their training and application of statistics and empirical methods over the last 2 decades, even establishing a highly rated research journal called *Empirical Software Engineering*. Nevertheless, there are opportunities where the more sophisticated empirical training of I-O psychologists can be teamed with subject matter experts to advance the quality of research on Lean/Agile practices at both the team and individual levels. Individual differences in developing software are enormous (Curtis, Sheppard, Kruesi-Bailey, Bailey, & Boehm-Davis, 1989) and often overwhelm the main effects of practices being experimentally manipulated. I-O psychologists are versed in experimental and statistical methods for managing the impact of individual differences.

https://doi.org/10.1017/iop.2019.85 Published online by Cambridge University Press

Domain knowledge

Another hurdle facing I-O psychologists seeking to study Lean or other improvement practices, especially in cognitively complex disciplines, is lack of domain knowledge. In many domains, at least a basic knowledge of processes, methods, materials, products, services, and tools is important for understanding the factors to study and how to define appropriate measures (Curtis, Sheppard, Milliman, Borst, & Love, 1979). For instance, measuring performance by the number of computer instructions (lines of code) developers write will entangle one in endless debates about whether lines of code is a legitimate measure of individual performance or Lean/Agile effectiveness. The best designed and most cheaply maintained software is frequently smaller, and software can be produced more rapidly if little attention is given to its quality. These conundrums are not always apparent to domain newcomers.

Another challenge requiring domain knowledge involves testing theory. Some years ago, I was asked to review a proposed research agenda on software teams. I pointed out that several critical variables had not been included. I was sternly informed that unlike industrial research, academic researchers must test theory, and the theory did not include those variables. I suggested that a theory that fails to account for factors controlling the most variation in performance is not a helpful theory. A year and a half later the researcher was dismayed to find inconclusive results and admitted that variation from other factors had overwhelmed the data. Psychological theories and research results concerning the benefits and effects of Lean practices will be enhanced by deeper engagement with the domains in which Lean is applied.

Going native

I-O psychologists have made contributions to Lean and other improvement practices in industry. For instance, an I-O psychologist working at the Software Engineering Institute at Carnegie Mellon University led development of the Capability Maturity Model (CMM; Paulk, Weber, Curtis, & Chrissis, 1995), which has become the de facto global standard for evaluating the capability of a software development organization. In some cases, I-O psychologists embedded themselves into the domain and over time ceded their identity as psychologists. Those conducting research frequently publish in journals relevant to their chosen domain rather than I-O journals in order to expose their results to the audience that can best apply them. Sometimes this immersion was necessary to develop deep understanding of the primary factors affecting domain processes and outcomes. In other cases, improvement recommendations and research results were accepted as more credible because they were perceived as coming from a knowledgeable source within the domain.

The choice to "go native" depends on which community an I-O psychologist most wants to affect. Thus, the fact that articles about Lean and other improvement methods appear infrequently in psychological journals does not mean that I-O psychologists are not engaged in research and practice on Lean and related improvement techniques. It may mean that some I-O psychologists have shifted their audience to the ultimate customer of their behavioral advice. Lean practices and other improvement methods may seem just outside the comfort zone of I-O psychologists, but once engaged in a domain of application, their contributions can be substantial.

References

Balzer, W. K., Brodke, M. H., Kluse, C., & Zickar, M. J. (2019). Revolution or thirty-year fad? A role for I-O psychology in Lean management. *Industrial and Organizational Psychology: Perspectives on Science and Practice*, **12**(3), 215–233.

Bowers, C., Salas, E., & Jentsch, F. (2006). *Creating high tech teams*. Washington, DC: APA.

Brechner, E. (2015). *Agile project management with kanban*. Redmond, WA: Microsoft Press.

Curtis, B. (1981, Ed.). *Human factors in software development*. Washington, DC: IEEE Computer Society.

Curtis, B., Sheppard, S. B., Kruesi-Bailey, V., Bailey, J., & Boehm-Davis, D. (1989). Experimental evaluation of software specification formats. *Journal of Systems and Software*, **9**(2), 167–207.

https://doi.org/10.1017/iop.2019.85 Published online by Cambridge University Press

Curtis, B., Sheppard, S. B., Milliman, P., Borst, A., & Love, T. (1979). Measuring the psychological complexity of software maintenance tasks with the Halstead and McCabe metrics. *IEEE Transactions on Software Engineering*, **5**(2), 96–104.

Curtis, B., Soloway, E., Brooks, R., Black, J., Ehrlich, K., & Ramsey, H. R. (1986). Software psychology: The need for an interdisciplinary program. *Proceedings of the IEEE*, **74**(8), 1092–1106.

Meyer, B. (2014). *Agile! The good, the hype, and the ugly*. Zurich, Switzerland: Springer International Switzerland.

Mohrman, S. A., Cohen, S. G., & Mohrman, A. M. (1995). *Designing team-based organizations*. San Francisco, CA: Jossey-Bass.

Paulk, M., Weber, C., Curtis, B., & Chrissis, M.B. (1995). *The capability maturity model: Guidelines for improving the software process*. Reading, MA: Addison Wesley.

Poppendieck, M., & Poppendieck, T. (2003). *Lean software development: An agile toolkit*. Boston, MA: Addison-Wesley.

Poppendieck, M., & Poppendieck, T. (2007). *Implementing Lean software development*. Boston, MA: Addison-Wesley.

Reinertsen, D. G. (2009). *The principles of product development flow: Second generation Lean product development*. Redondo Beach, CA: Celeritas Publishing.

Cite this article: Curtis, B. (2019). Applying Lean to cognitively complex work. *Industrial and Organizational Psychology* **12**, 272–276. https://doi.org/10.1017/iop.2019.85

https://doi.org/10.1017/iop.2019.85 Published online by Cambridge University Press