The Rise and Evolution of Agile Software Development

Rashina Hoda, University of Auckland

Norsaremah Salleh, International Islamic University Malaysia

John Grundy, Monash University

// This article provides a historical overview of agile's main focus areas and a holistic synthesis of its trends, their evolution over the past two decades, and agile's current status and likely future. //

FPO

ORIGINALLY, COMPUTER SOFTWARE was written in an ad hoc manner. The programmers often had no formal training but great domain knowledge and aptitude, most commonly using large-scale nonnetworked machines and lacking a common set of principles and practices. This process was really more akin to a cottage industry than an engineering discipline.

Subsequently—owing to all the expected problems of such an approach—software engineering was developed as a discipline to provide engineering rigor to the profession. In the '70s, '80s, and early '90s, the growth of software systems, range of domains of applications, number of developers, advent of the web, and diverse range of challenging software engineering problems resulted in a set of printciples, methods, practices, and tools to assist the engineering of such systems.

In this move to complex processes, project management, tools, process and project measurement, documentation, and other supporting practices, the human side of engineering software was seen by many to have become, or to certainly be becoming, lost in mainstream software development. Emerging in the late 1990s in response to the then-prevalent complex methods, agile methods offered disciplined yet lightweight processes while placing human effort and experience at the core of software development, through its central focus on people and interactions.[1] Agile methods retain the rigor of engineering processes and best practices while better helping both stakeholders and software engineers build, deploy, and maintain complex software.

Agile has now become a major software engineering discipline in both practice and research. Formally introduced through a set of four core values and 12 principles laid out in the Agile Manifesto,[2] agile is now the mainstream software development method of choice worldwide.[1]

Agile in Practice

The latest State of Agile survey, the largest and longest-running survey of its kind, reports 97% of respondents' organizations practicing agile somewhere within their organization in 2018,[1] compared to 84% in the first survey in 2007.[3] Likely to be completed by those predisposed to agile, the latest survey also showed 52% having all or more than half of their teams practicing agile.

Scrum increased its prominence as the most popular agile method, from 40% reported in the first survey in 2007 to 56% on its own and 70% when combined with other methods in 2018. At the same time, Extreme Programming (XP) lost ground from being the second most popular method (23%) to being used in combination with Scrum at 6%. Meanwhile, Kanban on its own (5%) and in combination with Scrum (Scrumban, 8%) replaced DSDM (dynamic systems development method, 8%). In another new development, 71% of organizations report planning or investing in DevOps initiatives now.

In the first survey, reported concerns with adoption centered around lack of up-front planning, documentation, and predictability, and loss of management control. In the 2018 survey, the top challenges reported in adopting and scaling agile organizational culture were general organizational resistance to change and inadequate management support.

One of the most interesting findings is that an overwhelming 84% of organizations are still maturing in their agile practice. This highlights continued opportunities for agile research on the challenges of agile adoption and practice.

Agile Research

The phenomenal growth of agile practice is mirrored by agile research becoming a significant subdiscipline of software engineering in the last two decades and continuing today. In April 2018, a search for the keywords "agile software development" in Google Scholar for a period up to 2001 produced just over 13,000 results. The same search led to over 260,000 results for today.

Agile research has featured prominently in many premier software engineering periodicals, including *IEEE Transactions on Software Engineering*, *IEEE Software*, *Empirical Software Engineering*, *Journal of Systems and Software*, and *Information and Software Technology*. Agile research has also been published in flagship software engineering conferences such as the International Conference on Software Engineering (ICSE) and Foundations of Software Engineering (FSE) and in numerous other reputable conferences. In addition, the rise and sustained growth of agile research has been chronicled by 19 years of the International Conference on Agile Software Development (XP) and 15 years of the North American Agile Conference (Agile), two of the largest dedicated annual agile conferences, and by numerous regional agile conferences and events around the world.

We conducted a research retrospective in the form of a tertiary study[4] of 28 systematic literature reviews and mapping studies, capturing two decades of agile research.[5] We identified 10 key agile research areas: agile adoption, agile methods, agile practices, human and social aspects, agile and GSE (global software engineering), agile and usability, agile and CMMI (Capability Maturity Model Integration), organizational agility, agile and embedded systems, and software product line engineering, summarized in Figure 1.

The agile-practices area, which covered topics such as test-driven development, metrics, effort estimation, and requirements, was the most significant research area, with seven systematic reviews. Agile and

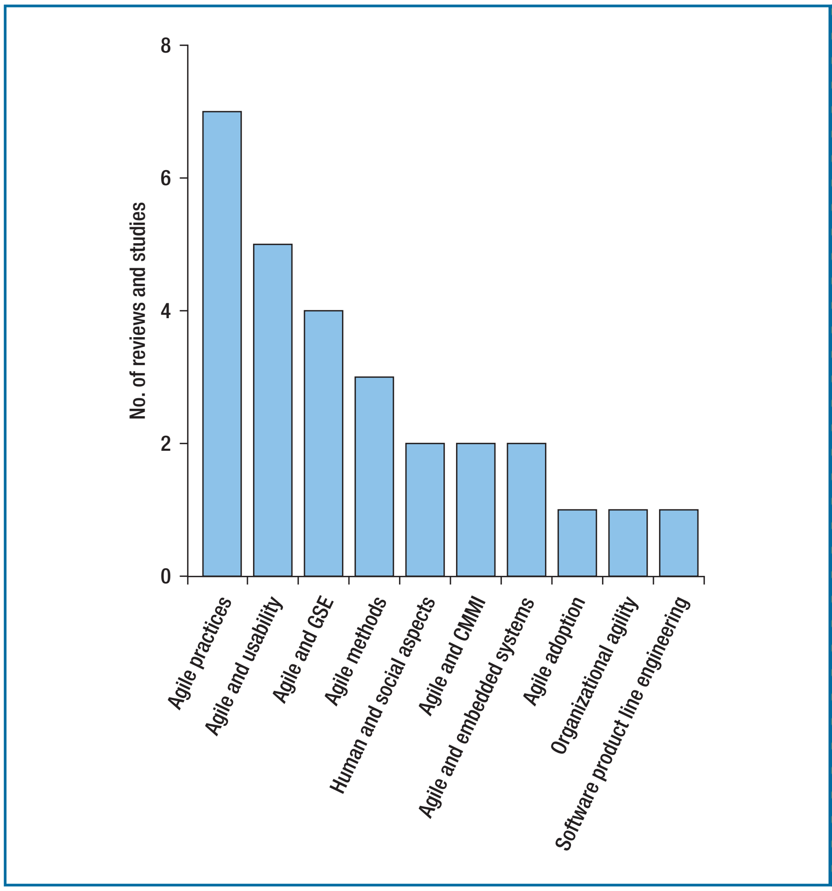

FIGURE 1. Systematic literature reviews on agile topics.[5] GSE = global software engineering, and CMMI = Capability Maturity Model Integration.

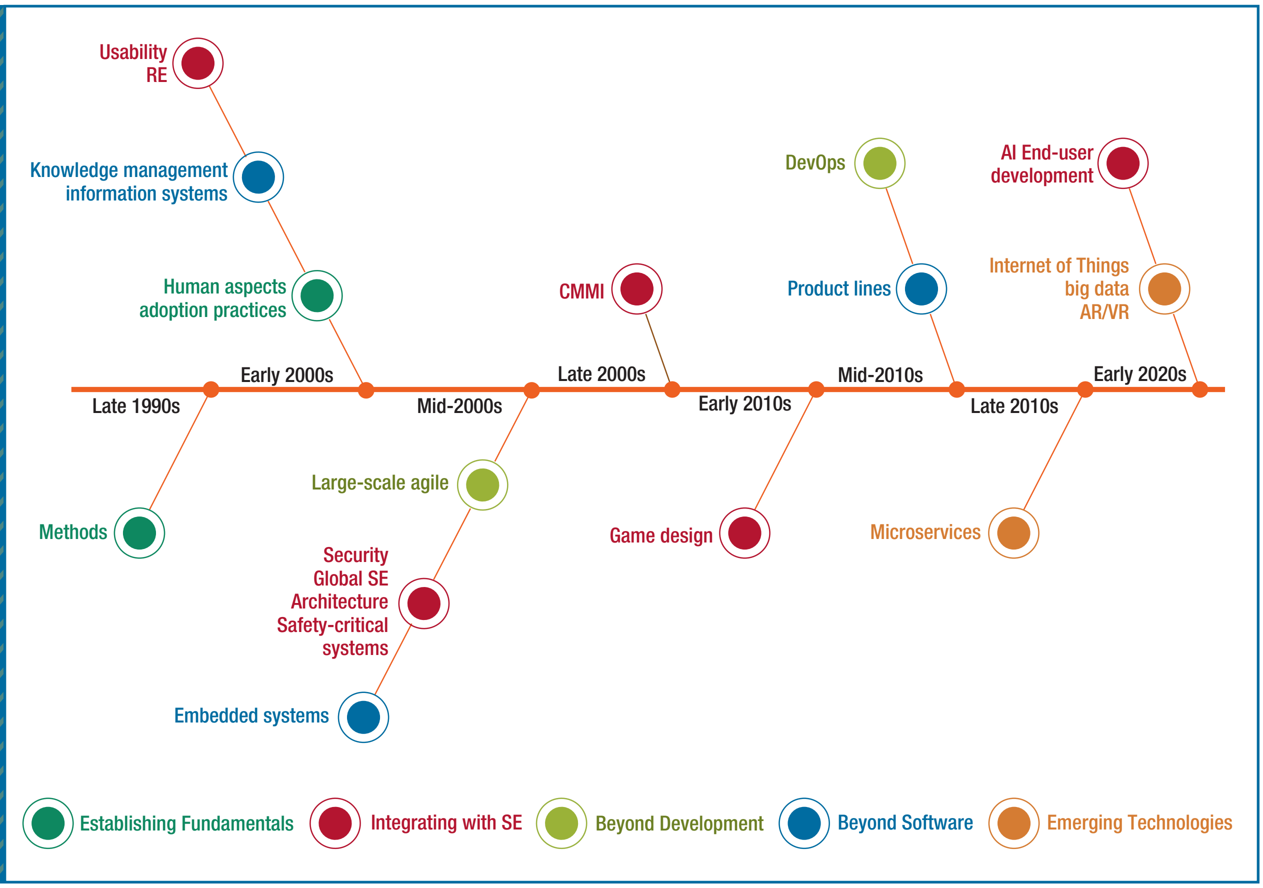

FIGURE 2. The emergence of trends in agile software development, based on the first relevant publications in the IEEE and ACM digital libraries. SE = software engineering, RE = requirements engineering, and AR/VR = augmented reality or virtual reality.

usability had the second highest number of reviews, five, and focused on topics such as integrating the user experience with agile. Given the role of human–computer interaction in maintaining a focus on engineering for people and its synergies with agile software development, this is not surprising. This was followed by agile and GSE, with four reviews.

Agile education, an active and vibrant research area, is not included in our study, as we focus on industrial research. Another significant agile research area is pair programming, one of the most popular XP practices. That area deserves a secondary review to collate and present the numerous research studies under that banner.

Agile Evolution

There is little consensus among industrial reports and sources on the emergence of particular trends in agile software development over time. However, the first relevant publications in each of these areas are well documented in some of the largest publication archives and digital libraries—e.g., the IEEE and ACM digital libraries. We used the information from those digital libraries to devise a timeline illustrating the rise and evolution of agile software development (see Figure 2).[1–4] Indicative papers charting this timeline can be found at http://dx.doi.org/10.21227/H2T08T.

The early days of agile saw exploration of fundamental agile concepts such as agile methods,[6] agile adoption, agile practices, and the human and social aspects, combined into the Establishing Fundamentals trend (see the dark-green elements in Figure 2). For example, the single largest review (with 333 papers) was on the role of communication, a human and social aspect fundamental to agile software development. With the latest State of Agile survey reporting 84% of organizations to be still maturing in agile,[1] many of these fundamental issues continue to be relevant in practice. Similarly, the agile research community continues to call for establishing firmer theoretical foundations of agile research, keeping this trend very much alive.

The Integrating with Software Engineering trend (see the magenta elements in Figure 2) saw synergies explored between agile software development and some of the more well-established software engineering concepts and subdisciplines, such as usability, requirements engineering, software security, GSE, software architecture, and safety-critical systems, in the early and mid-2000s. Other new topics in this trend emerged over time, such as integrating agile and CMMI in the late 2000s and agile and game design in early 2010s. Most of these topics have continued to be popular, and some—in particular, security and safety-critical systems—have witnessed a strong surge in renewed interest with recent technological advancements such as blockchains and cryptocurrencies.

Small, colocated teams, with an onsite or easily available customer, an emphasis on programming and early testing, and frequent feedback on iterative delivery of working software, marked the original agile sweet spot.[7] The mid-2000s saw agile step outside its comfort zone, scaling beyond the confines of small development teams into large-scale agile, applying hybrids of agile software development at the intrateam level and traditional planning approaches at the interteam level.

Once again, in the mid-2010s, agile ventured Beyond Development to acknowledge operations alongside development, through DevOps. Continuous delivery and continuous feedback from users to developers would seem a natural fit. However, many technical, sociotechnical, and organizational challenges present themselves. When and how should customer feedback be captured and actioned, and when should changes be rolled out? What about software deployed across different organizations and user groups with different requirements? And when software infrastructure changes significantly, how is continuously deployed software effectively tested? What is the impact of DevOps transformations on agile practices?

Another significant trend involved extending agile Beyond Software, its original domain, and into related disciplines such as knowledge management and information systems in the early 2000s. Given its central focus on human and social aspects, agile has brought the complementary disciplines of software engineering and information systems closer than ever before. Software engineering has gained from the theoretical robustness of information systems research, and the information systems field has gained from the practical relevance of software engineering studies on agile topics.

Agile also spread into closely related areas such as embedded systems, starting in the mid-2000s, and product line engineering, starting in the mid-2010s. Traditionally, these domains have had their own processes, practices, measurements, and team cultures. Embedded systems traditionally have been dominated by engineers using waterfall-style processes heavy on planning, documentation, measurement, and model-driven tool support.

Applying agile software development philosophies, practices, and cultures to these domains is challenging. And yet, agile has advanced into these domains rapidly. For example, this rapid advance is occurring in the automotive industry, as it becomes more and more software-intensive with the advent of autonomous-vehicle technologies.

Finally, in the late 2010s, we see an interest in exploring the tensions and synergies between agile and microservices. Emerging microservice-based architectures take software by composition to a new level. This is impacting software design and deployment and raises questions such as, how does a team balance its agile practices with emerging microservice architectures that require some level of design up front?

Agile in the Future

With current advancements in technologies such as the Internet of Things (IoT), a wide range of devices are being integrated into systems, vast amounts of big data are becoming available for analysis, various augmented- and virtual-reality systems are being developed, and intelligent solutions are increasingly expected. At the same time, these emerging technologies have renewed interest and opened new possibilities in exploring the full potential of not-so-new paradigms such as AI and end-user development,[8] going forward in the 2020s.

We predict a strong role for agile software development in partnering with and enabling the Emerging Technologies trend in the foreseeable future. We expect a number of questions to be explored:

- How will agile practices enable AI-based software engineering? In the last three years, there has been a large increase in publications on new AI-based software tools. Conversely, how can AI be used to augment agile software development?

- Can agile improve data analytics and data sciences practices in the way it has improved software engineering? Is there an agile approach to data science that leverages practices similar to those of agile software engineering?
- To fulfil the demand of the IoT, to what extent can agile methods revolutionize the IoT industry? How do hardware, embedded, creative, visual, source, touch, and other interface designers work effectively with, or indeed within, agile software development teams? IoT solutions may be composed from hardware and software components. How do we produce more agile hardware solutions?
- Security continues to be a major concern for developers and users. While agile practices and continuous-deployment approaches theoretically allow for quick fixes of emerging security issues, extensive security testing before deployment is increasingly being required. Similarly, zero-day security threats can't be fully designed or tested for, but an agile fix may not be acceptable in many circumstances, either. How do agile methods ensure that security requirements are continually met?
- How can agile processes support the development of safety-critical systems in increasingly software-intensive autonomous vehicles, software-defined networking, and robotics development and integration?
- End-user development of complex software, whether by coding, configuration, composition, or a mixture, is likely to continue to increase. Can agile practices support the development of software by non-technical experts who nonetheless want to quickly and effectively improve and deploy parts of the software systems they use? Where does an agile software development team end and end-user developers of their own (parts of a) software system begin?
- How do we successfully leverage agile across multiple emergent technology domains and practices? For example, what does "agile, secure DevOps for data-intensive intelligent systems" mean for researchers and practitioners?

Since its inception in the late 1990s, agile software development has come to dominate the latter half of the past 50 years of software engineering. It started off with Establishing Fundamentals such as agile adoption, methods, and practices, and human and social aspects. Then it moved on to Integrating with Software Engineering topics and subdisciplines such as usability, requirements engineering, GSE, software architecture, CMMI, and game design. Of late, agile development has seen renewed interest in security and safety-critical systems and seems likely to move into exploring synergies with AI and end-user development.

Research has been directed at understanding how agile is made to work in practice within and alongside these preestablished software engineering paradigms. Barriers, areas of conflict, synergies, strategies, and workarounds have been researched and presented. Moving further out of its original comfort zone of small, colocated teams, agile has spread Beyond Development into DevOps implementations and large-scale agile on the business and enterprise levels.

After more than two decades of practice, organizations consider themselves still maturing in successfully deploying, improving, and contextualizing their agile practices to their teams, customers, and specific project conditions. Researchers continue to study these issues and to help practitioners comprehend and address them. Another fundamental issue, that of managing change within a process that actively promotes embracing change, demands further inquiry.

However unsure as we may feel about our collective maturity in agile software development, software engineers are indeed looked upon as the experts in agile practice by those in disciplines Beyond Software, such as embedded systems and product lines. Agile practitioners can assist in agile transformations outside of software development—e.g., in human resources, sales and marketing, project management, and R&D. They can do this by abstracting out the lessons learned from agile transformation in software teams, applying them to new contexts, and helping adapt agile to fit new contexts.

Finally, peeking into the future, much give and take can be expected between agile and Emerging Technologies such as the IoT, augmented and virtual reality, big data services, paradigms such as AI, and end-user development.

References

1. *12th Annual State of Agile Survey Report*, VersionOne, 2018; http://stateofagile.versionone.com.
2. M. Fowler and J. Highsmith, "The Agile Manifesto," *Software*

Development, vol. 9, no. 8, 2001, pp. 28–35; http://agilemanifesto.org.
3. *Survey: "The State of Agile Development*," VersionOne, 2007; http://stateofagile.versionone.com.
4. B.A. Kitchenham and S. Charters, *Guidelines for Performing Systematic Literature Reviews in Software Engineering*, tech. report EBSE-2007-01, Keele Univ. and Univ. of Durham, 2007.
5. R. Hoda et al., "Systematic Literature Reviews in Agile Software Development: A Tertiary Study," *Information and Software Technology*, vol. 85, 2017, pp. 60–70.
6. M. Aoyama, "Web-Based Agile Software Development," *IEEE Software*, vol. 15, no. 6, 1998, pp. 56–65.
7. D. Reifer, F. Maurer, and H. Erdogmus, "Scaling Agile Methods," *IEEE Software*, vol. 20, no. 4, 2003, pp. 12–14.
8. J. Segal and C. Morris, "Developing Scientific Software," *IEEE Software*, vol. 25, no. 4, 2008, pp. 18–20.

ABOUT THE AUTHORS

RASHINA HODA is a senior lecturer (associate professor) and the founder of the Software Engineering Processes Tools and Applications research group at the University of Auckland. Her research interests include agile software development, cooperative and human aspects of software engineering, grounded theory, and serious-game design. Hoda received a PhD in computer science from Victoria University of Wellington. Contact her at r.hoda@auckland.ac.nz.

NORSAREMAH SALLEH is an associate professor in the Department of Computer Science, International Islamic University Malaysia. Her research interests include empirical software engineering (SE), evidence based SE, the human and social aspects of SE, and computer science and SE education. Salleh received a PhD in computer science from the University of Auckland. Contact her at norsaremah@iium.edu.my.

JOHN GRUNDY is the senior deputy dean of Monash University's Faculty of Information Technology. His research interests include automated software engineering, software tools, human-centric software engineering, visual languages, software architecture, software security engineering, and user interfaces. He's a Fellow of Automated Software Engineering and of Engineers Australia. Contact him at john.grundy@monash.edu.