In IEEE Software - 50th Anniversary Issue (c) IEEE 2018

THE RISE AND EVOLUTION OF AGILE SOFTWARE DEVELOPMENT

Authors: Rashina Hoda, Norsaremah Salleh, John Grundy

ABSTRACT. Agile Software Development has dominated the second half of the past 50 years of Software Engineering. Retrospectives, one of the most common agile practices, enables reflection on past performance, discussion on current progress, and charting forth directions for future improvement. The burgeoning popularity of Agile as the software development model of choice and a significant research sub-domain of software engineering demands a retrospective of its own. In this article, we provide a historical overview of Agile's main focus areas and a holistic synthesis of its trends, their evolution over the past two decades, current status, and forecast from these the likely future of agile software development.

Rise of Agile

Originally computer software was written in an ad-hoc manner, often by those without any formal training but with great domain knowledge and aptitude, most commonly using large-scale non-networked machines, lacking a common set of principles and practices, and was really more akin to a cottage industry than an engineering discipline. Subsequently - due 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 world wide web, and diverse range of challenging software engineering problems resulted in a set of principles, 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 by and for people was seen by many to have been, or was certainly becoming, lost in mainstream software development. Emerging in the late 1990s in response to the then prevalent complex methods, Agile methods offered disciplined yet light-weight processes while placing the human effort and experience at the core of software development through its central focus on *people and interactions*.[2] Agile methods retain the rigor of engineering processes and best practices while better supporting both stakeholders and software engineers to build, deploy and maintain complex software.

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

State of Agile

Agile in Practice

The latest *State of Agile* survey,[2] the largest and longest running survey of its kind, reports 97% of respondents' organizations practicing agile anywhere within their organization in 2018 compared to 84% in the first survey in 2007.[3] The latest report also showed 52% having all or more than half 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 (8%). In another new development, 71% of organizations report planning or investing DevOps initiatives now.

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

One of the most interesting findings is that an overwhelming 84% of organizations are "still maturing" in their agile practice, highlighting 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 sub-discipline of software engineering in the last two decades and continues today. A search for keywords "agile software development" for a period up to 2001 produces just over 13,000 results. The same search leads to over 260,000 results in Google Scholar today (as of April 2018).

Agile research has featured prominently in premier software engineering journals including IEEE Transactions on Software Engineering, IEEE Software, Empirical Software Engineering, Journal of Systems and Software, Information and Software Technology, and many more. In addition to being published in flagship software engineering conferences such as ICSE and FSE, and numerous reputed conferences, 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 numerous regional agile conferences and events around the world.

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

Agile practices covered topics such as test-driven development, metrics, effort estimation, and requirements, was the most significant research area with seven systematic reviews. *Agile and usability* included the second highest number of reviews, five, and focused on topics such as integrating user experience with agile. Given the role of HCI 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 Global Software Engineering* (GSE) with four reviews in this area.

Agile education is an active and vibrant research area, 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, which deserves a secondary review to collate and present the numerous research studies under this banner.

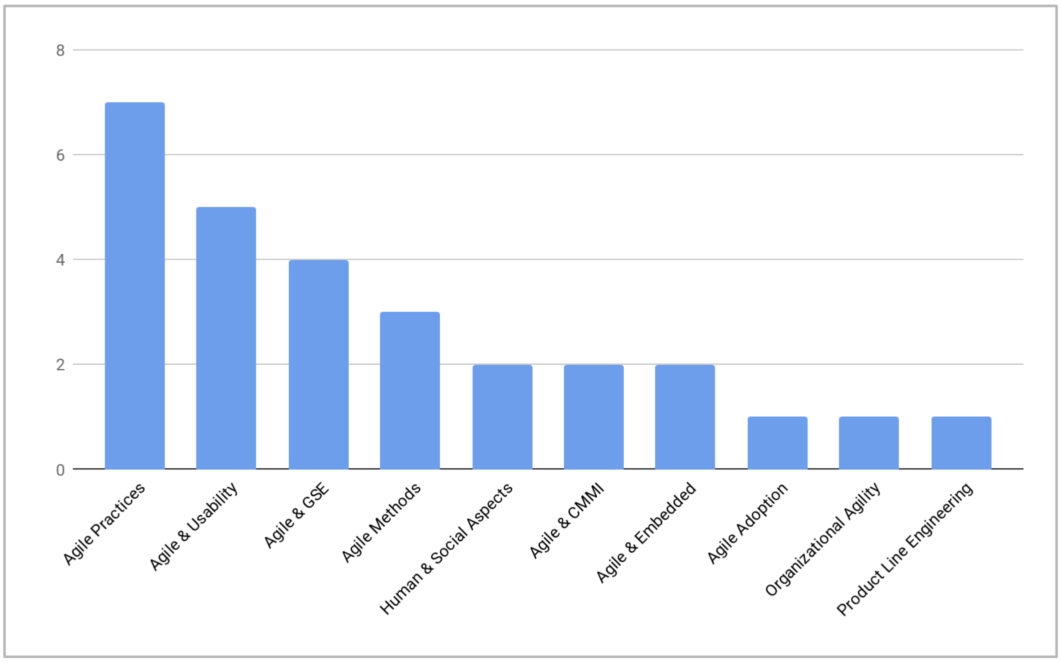

Figure 1. Systematic Literature Reviews (SLRs) on Agile Topics [5]

Sidebar: Trends in Agile Software Development

Agile Evolution

While there is little consensus among industrial reports and sources on the emergence of particular trends in agile software development over time, the first relevant publications in each of these areas is well documented in some of the largest publication archives and digital libraries (e.g. IEEE and ACM) which we have used to present the ***emergence of trends in agile software development*** timeline (Figure 2)[1–4]. Indicative papers charting this timeline can be found at https://sites.google.com/view/agiletimeline.

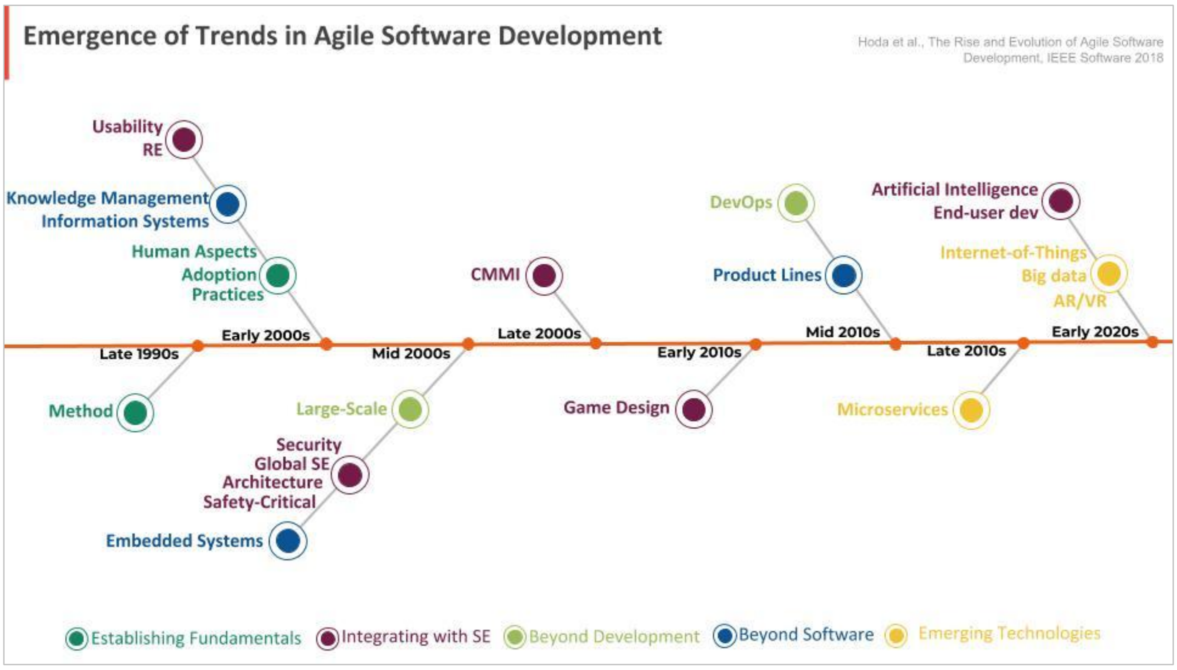

Figure 2. The Emergence of Trends in Agile Software Development
Based on first relevant publications in IEEE and ACM digital libraries

---

The early days of agile saw exploration of fundamental agile concepts such as *agile methods*[7], *agile adoption*, *practices*, and *human and social aspects*, combined into the trend ***establishing fundamentals*** *(dark green elements in the timeline, 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[2] reporting 84% organizations to be "still maturing" in agile, 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.

Integrating agile software development with some of the more well-established software engineering concepts and sub-disciplines also emerged as a trend in the early and mid-2000s. This trend, ***integrating with software engineering*** *(magenta elements in timeline, Figure 2)*, saw synergies between explored between agile and; *usability, requirements engineering, software security, global software engineering, software architecture, and safety-critical systems*. Other new topics in this trend emerged over time, such as integrating agile and *capability maturity models* 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 blockchain and cryptocurrencies.

Small, co-located teams, with an on-site 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".[6] 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 intra-team 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, socio-technical and organisational challenges present themselves. When and how should customer feedback be captured, actioned, and changes rolled out? What about software deployed across different organisations 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 early 2000s. Given its central focus on human and social aspects, agile has brought the complementary disciplines of software engineering and information systems closer like never before, SE gaining from the theoretical robustness of IS research and IS gaining from the practical relevance of SE studies, on agile topics.

Agile also spread into closely related areas such *embedded systems* starting in mid-2000s and *product line engineering* starting mid-2010s. Traditionally these domains have had their own processes, practices, measurements and team cultures. Embedded systems have traditionally 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 areas rapidly as the automotive industries become more and more software-intensive, for example, 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, impacting software design and deployment and raising questions such as how does a team balance its agile practices with emerging micro-service architectures that require some level of design-up-front?

Agile in the Future

With current advancements in technologies such as *internet-of-things (IoT),* a wide range of devices are being integrated into systems, vast amounts of *big data* is 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 *artificial intelligence* and *end-user development*[8] going forward in the 2020s.

We predict a strong role of agile software development in partnering with and enabling these ***emerging technologies*** in the foreseeable future, expecting a number of questions to be explored.

- How will agile practices enable AI-based software engineering? There has been a large increase in the last 3 years of publications on new AI-based software tools. How can AI be used to augment agile software development?
- Can agile improve data analytics and data sciences practices in the way it has software engineering? Is there an Agile approach to Data Science leveraging similar practices to those of Agile software engineering?

- To fulfill the demand of 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 to 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 increasing 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 security requirements are continuously 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 - e.g. what does "agile, secure DevOps for data-intensive intelligent systems" mean for researchers and practitioners?

Conclusion

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. Starting with ***establishing fundamental*** concepts such as agile adoption, methods, practices, and human and social aspects, it moved on to ***integrating with software engineering*** topics and sub-disciplines such as usability, requirements engineering, global software engineering, software architecture, CMMI, game design; with a renewed interest in security and safety-critical systems of late, and looking to move into exploring synergies with artificial intelligence and end-user development going forward. Research has been directed at understanding how agile is made to work in practice within and alongside these pre-established software engineering paradigms. Barriers, areas of conflict, synergies, strategies and workarounds were researched and presented. Moving further out of its original comfort zone of small, co-located teams, agile spread ***beyond development*** into DevOps implementations and large-scale agile on the enterprise level.

After more than two decades of practice, organisations consider themselves "still maturing" in successfully deploying, improving, and contextualizing their agile practices to their teams, customers, and specific project conditions, and researchers continue to study and assist practitioners comprehend and address these issues. 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. HR, sales and marketing, project management, research and development by abstracting out the lessons learnt from agile transformation in software teams and 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 internet-of-things (IoT), AR/VR, big data services and paradigms such as artificial intelligence, and end-user development in the foreseeable future.

References

1. M. Fowler and J. Highsmith. The agile manifesto, Software Development, 9(8), 28-35, 2001. Available: http://agilemanifesto.org/ Last accessed 24th Jan 2018.
2. VersionOne. 12th Annual State of Agile Survey Report, Available http://stateofagile.versionone.com/ Last accessed 2nd May 2018.
3. VersionOne. The State of Agile Development (First Survey Report), Available http://stateofagile.versionone.com/ Last accessed 24th Jan 2018.
4. B.A. Kitchenham and S. Charters. Procedures for Performing Systematic Literature Review in Software Engineering, EBSE Technical Report version 2.3, EBSE-2007-01, Software Eng. Group.
5. R. Hoda, N. Salleh, J. Grundy, H.M. Tee. Systematic literature reviews in agile software development: A tertiary study, Information and Software Technology, vol. 85, pp. 60 – 70, 2017.
6. D. Reifer, F. Maurer, H. Erdogmus. Scaling Agile Methods IEEE Software, July/August 2003 IEEE, 2003
7. M. Aoyama. Web-based agile software development. IEEE software, 15(6), 56-65, 1998.
8. J. Segal, C. Morris. Developing scientific software. IEEE software, 25(4), 18-20, 2008.

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

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

JOHN GRUNDY is Senior Deputy Dean of the Faculty of Information Technology at Monash University. His research interests include automated software engineering, software tools, human-centric software engineering, visual languages, software architecture, software security engineering and user interfaces. He is Fellow of Automated Software Engineering and Fellow of Engineers Australia.