Agile Development: The Promise, the Reality, the Opportunity

Jon W. Beard[1], Veda C. Storey[2], Binny Samuel[3], Roman Lukyanenko[4], Anna Wiedemann[5], David Schuff[4], Shawn Ogunseye[6], Zohra Islamzada[7], Fereshta Islamzada[4]

[1] *Ivy College of Business, Iowa State University, Ames, IA 50011-2027 USA (Corresponding Author)*
[2] *Robinson College of Business, Georgia State University, Atlanta, GA 30303 USA*
[3] *Linder College of Business, University of Cincinnati, Cincinnati, OH 45221 USA*
[4] *McIntire School of Commerce, University of Virginia, Charlottesville, VA 22903 USA*
[5] *ZHAW School of Management and Law, Zurich University of Applied Sciences, 8401 Winterthur, Switzerland*
[6] *Computer Information Systems, Bentley University, Waltham, MA 02452 USA*
[7] *Freddie Mac, 8200 Jones Branch Dr., McLean, VA 22102 USA*

Abstract

Agile Development has been an integral part of project management and product development since its formal introduction by the *Agile Manifesto* in 2001. Subsequently, Agile has rapidly gained in popularity, leading to significant improvements in on-time delivery and managing costs, as well as successful delivery of the required scope for information systems (IS) projects. Agile has even moved beyond the IS domain into other business applications. This success may be tempered somewhat by the gaps in the Agile Development process that still exist, such as a lack of complete understanding of the process or the culture changes necessary to achieve larger benefits and the complexity of integrating Agile-developed products into existing IT infrastructure. However, there is also great promise in extending Agile by incorporating new tools and concepts and applying Agile development to novel and emerging problem domains. This paper proposes a Framework for Agile Development that can be used to explore existing results from Agile development, as well as to identify future possibilities and emphasize the importance of teaching Agile development in order for it to have further influence on practice.

Keywords

Agile development, Framework for Agile Development, Agile Development Framework, Agile teaching, Agile Manifesto, information systems, complex applications, emerging technology, Agile analytics, conceptual modeling, DevOps

1. Introduction

Agile Development is an approach to software and information systems (IS) development that emphasizes collaboration with users though an iterative and incremental development process [1]. Agile was created in response to the challenges of the traditional approach, known as the *Waterfall* method, which was often plan-driven and characterized as a predictive approach. The latter approach required complete, accurate, and approved specifications before development began, and restricted design changes as the project progressed [2].

Today, we are experiencing an expanded use of IS applications in both professional and personal activities. Systems developers continuously strive to embrace the complexities of today's world, acknowledging the ever-changing technology landscape, which is being accessed by users both within and outside their organizations. The result has made the need for, and positioning of, Agile development even more pronounced. Since Agile has become widely adopted in practice, it is important to consider how to research and teach this important software and information systems development approach [3].

Agil-ISE24: 3rd Intl. Workshop on Agile Methods for Information Systems Engineering, June 3, 2024, Limassol, Cyprus
EMAIL: jwbeard@iastate.edu, vstorey@gsu.edu, binny.samuel@uc.edu, romanl@virginia.edu, anna.wiedemann@zhaw.ch, david.schuff@virginia.edu, sogunseye@bentley.edu, zzohrai28@gmail.com, acu6bg@virginia.edu

In this paper, we examine the *promise* of Agile as originally envisioned in the Agile Manifesto [4], the *reality* and challenges of Agile being applied, and the *opportunity* to extend the influence of Agile in research, teaching, and practice. To do so, we propose a Framework for Agile Development (hereafter called the Agile Development Framework), which is intended to capture the progression of Agile and to identify future possible research and teaching areas. Figure 1 portrays this Framework. It is intended to provide a general and useful way to understand a phenomenon [5]. The Framework was motivated by the experiences of Agile educators, a review of the current literature, industry reports, consultations with Agile experts, and suggestions of emerging topics as Agile trends for future exploration. In the sections that follow we discuss the three main categories of the Framework—the Promise, the Reality, and the Opportunity for Agile—each of which identifies important topics that deserve to be explored. These guide the remainder of the discussion and presentation of the concepts required for advancing Agile Development. Given the constraints of the presentation format for this paper and the related Workshop we explore only some of the topics in this paper. Our contribution with this framework is to recognize new emerging technologies and increasingly important real-world and global problems that Agile can address as we believe Agile can be applied to a wide range of challenges today. We also seek to inspire the next generation of Agile practice (including educational settings where individuals are trained) to ensure its continued adoption in practice.

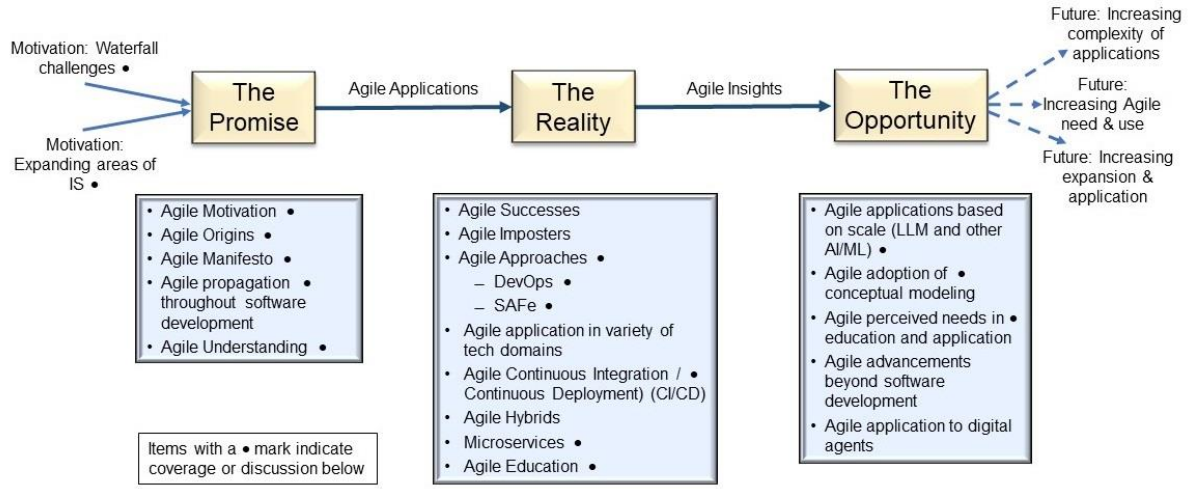

Figure 1. Agile Development Framework

2. The Promise of Agile Development

Many aspects of the Agile approach have been in use in systems development since the 1950s and increasingly applied in the 1990s [6]. For example, Spiral Development (1986), Rapid Application Development (1991), the Unified Process (UP) (1994), and Extreme Programming (XP) (1996), among others, all contained features or characteristics now attributed to Agile Development [7,8]. However, the writing and public release of the *Agile Manifesto* [4] has generally been regarded as the birth of *Agile Development* [9]. Prior to that time, traditional approaches to IS analysis, design, and development were most often described as top-down decomposition, structured (i.e., sequential), highly detailed, heavily documented, and meticulously managed, and, thus, were considered to be "predictive" and "heavyweight." The traditional approaches usually assumed functional requirements could be fully and accurately articulated before design activities began [10].

In contrast, Agile development approaches are a broad framework that can be characterized as 'lightweight' due to their less-tightly structured development approach and their emphasis on an evolving modular design through iterative and incremental practices. Agile encourages ongoing collaboration with users and dynamic innovation throughout the design, development, and implementation process. Scrum is the most popular type of Agile approach [1], with emphasis on accomplishing the development of a project in small, usually dedicated, teams. Further, the incremental, iterative nature of Agile specifically fosters regular self-reflection—opportunities for improved

collaboration, learning, and adaptation—within the team on how to improve the development process [4, 11].

While traditional methods thrive when developers understand the problem and have a clear vision of the solution or product to be developed, Agile is useful when it is difficult to articulate the full set of requirements or characteristics for a product that is being developed upfront, i.e., there is a need for ongoing product discovery [2]. Preferences and solution options change frequently, close collaboration and rapid feedback is needed from customers, complex problems have unknown solutions, and are ambiguous in scope. The result is the need for an incremental, iterative development process with learning from mistakes [1]. The product's details can be discovered through iterative design efforts with incremental improvements that progressively add value based on user feedback. The active involvement of the users is intended to lead to a better fit with user needs and preferences. A product design is selected, built, and released with refinements being made based on the feedback that is received from the users, followed by further incremental development, testing, another release to users, and future refinement. This process of continual experimentation, feedback (i.e., failing fast), and learning should ultimately lead to a more successful product. With the digitalization of organizations, rapid growth in demand, and growing expectations from consumers, the need for better and faster approaches to product discovery is continuing to grow in importance. This includes the development of novel products where there is scant knowledge about what an ideal solution looks like. (For more detailed descriptions and discussion of Agile, see [11, 12].)

3. The Reality of Agile Development

Agile development methods have become the status quo approach for developing business-critical software; the transition to Agile development approaches has been very successful with reports of as many as 95% of developers now using at least some Agile features in their digital product development efforts [11]. Recognizing the benefits of Agile development and delivery, other IS design and development approaches have also embraced greater flexibility. For example, in infrastructure, microservices have risen in popularity due in part to the iterative-release nature of software (e.g., continuous delivery) resulting from practices such as Agile [13, 14]. Microservices provide benefits to organizations that adopt it in terms of 1) independence in how multiple software teams can collaborate; 2) better IT alignment with the organization; and 3) flexibility in IT tools used on any given project [15]. Thus, not only is Agile having an impact in software and IS development, its iterative and flexible ethos are also positively impacting other areas of IS. In this section, we look at the current practice and teaching of Agile and identify current trends, gaps, and failures in Agile based on our review of prior research.

3.1. Agile accommodations for practice

There has been further development of Agile methods with new concepts being used in practice. As examples, the DevOps (Development and Operations) concept and large-scale Agile methods are the two leading efforts [16]. DevOps broadens the agile software development approach and includes software operations and ultimately leads to integration into the larger IT infrastructure. Consequently, the software delivery and maintenance process are now end-to-end Agile, closing the gap between software development activities and the handover to software operations. The DevOps team is responsible for the entire software delivery lifecycle. Typically, a DevOps team focuses on the delivery of one or more software products, displaying a cross-functional character by incorporating a business view (e.g., product owner), as well as software developers and operations staff. Consequently, classic project management methods reach their limits, as DevOps teams lack a defined start and end-date and require continuous budgeting, given that the scope now encompasses the development and provision of a software product throughout its complete lifecycle [16, 17].

Coordinating multiple ongoing projects and teams in organizations with Agile is challenging. Therefore, large-scale Agile extends Agile methods to multiple team settings. These frameworks are typically utilized in large organizations involved in large projects and/or multiple programs. Implementing large-scale agile frameworks, such as SAFe (Scaled Agile Framework for the

Enterprise), results in changes to organizational structures, processes, as well as cultural aspects of the organization [18]. Consequently, organizations often aim to achieve organization-wide digital transformations with the help of large-scale agile frameworks [19].

DevOps and large-scale Agile are just two examples of further enhancements of Agile methods. Both methods build upon and leverage the original agile ideas. An example is continuous integration (CI). CI aims to help software developers bridge the gap in software delivery. While this topic is not new to the Agile way of working, the introduction of the DevOps concept allows teams to implement CI in their software delivery lifecycle. Continuous learning and further development of Agile development [16] need to be incorporated into how we teach these skills. One possibility is to focus on the development of T-shaped skills because job roles are shifting from primarily being an emphasis on a specialist toward being a combined specialist in one area plus a generalist in another area [20].

3.2. Navigating gaps and failures in an Agile environment

The Agile approach has revolutionized software development by encouraging flexibility, collaboration, trial-and error (also called "interim mistakes" [1], and short iterative turnarounds. However, beneath some of the beneficial principles of Agile, prior research has identified hidden gaps that lead to miscommunication, a reduced sense of morale, and, too often, a lack of successful iterations [21].

One main failure is the misalignment between the development teams and stakeholders [22]. Although Agile promotes constant cross communication and intentional check-ins with customers these meetings can turn into conversations on additional requirements added to the initial task, i.e., *scope creep*. With differing priorities and a lack of understanding between the business perspective and a technical background this can cause confusion, irritation, and in turn delayed deployment dates.

Although the flexibility of an Agile environment can have many benefits (i.e., Agile teams are often expected to adjust quickly to changes in priorities and new requirements) it can be challenging for developers. For example, teams can be so immersed in one project, working with one programming language and building relationships with the customer that if priorities shift or development needs require a swift pivot in response to market demands, the developers are the ones feeling a lack of accomplishment since they must make the largest jump to an entirely new project that includes a whole new set of demands.

Furthermore, the rapid iterative nature of an Agile environment too often promotes unsuccessful code deployments, especially since not all development teams have fully automated processes. While automation can help streamline deployment processes, reduce manual overhead, and enable all members of the team to help with testing, it is not a prerequisite to being Agile. Therefore, there can be an incredibly large amount of variability and human error when working in any sort of technical environment. Teams that do not have automated testing must be able to detect when a story will need multiple peer reviews and must, therefore, set aside a substantial amount of time for this. However, when there are deliverables and deadlines that also need to be met, it is difficult to allocate time for every single test case before deployment. This often reduces the quality of the product being built and released. Even when teams have the facilities and time they need, timeboxing them to sprints and pursuing MVPs (Minimum Viable Products) may lead team members to choose easy or quick-to-implement solutions to problems in the present that may require additional resources to address later. This departure from the principle of delivering high-quality products initially may lead to greater technical debt and higher overall costs in the long run.

Despite these challenges Agile can be considered an inviting, transparent, and collaborative work environment. One significant benefit to adopting this methodology is the promotion of knowledge sharing. Planning meetings, retrospectives, and daily standups encourage members of the team to share and explain any issues they are having to the rest of the team. These activities promote reflection periods and the consideration of solutions for future issues that may arise. It provides a chance for the team to come together and learn from one another and promotes a bonding experience and a group mindset of continuous improvement.

While Agile environments may have flaws including miscommunication, low morale, and issues between iterations there are still many positive outcomes to adapting this methodology. There are

challenges but this process provides teams with the ability to collaborate, communicate, and motivate one another to optimize workflow which can lead to long-term success.

3.3. Fostering Agile methodologies in an educational environment

The structure of a classroom cannot compare to the unpredictability and rapid change of an Agile work environment. The transition from theoretical learning to practical application is already a challenge. Students accustomed to the structure and predictability of academic schedules and assignments often find themselves disoriented by the fluidity and rapid change inherent in Agile projects [23]. Exposing students to Agile methodologies in an educational environment will encourage students to iterate design ideas through sprints, embrace change, practice openness, increase collaboration, and improve their ability to pivot from one project to the next. As noted in the Agile Manifesto [4]: "Agile processes harness change for the customer's competitive advantage." Integrating Agile values into the education environment will help foster students who will be better able to acclimate to Agile practices.

While the demand for Agile developers and project managers is strong, an "active learning" approach to the Agile methodology within a classroom setting can be challenging, especially at the undergraduate level. Preparing students for the psychological and cultural realities of Agile is pivotal; learning about the Agile way of working early in one's educational career will help reduce the burden of having to shift mindsets while starting a career. Agile is as much about mindset as methodology, so preparation is essential for swift adoption. Breaking down conventional teaching styles and instead enlightening students on an atypical scholar mindset of fluidity, reflection, and constant communication. It is crucial for educational institutions to keep up with the trends to prepare students for success.

First, the iterative nature of the Agile methodology is time-consuming and difficult to operationalize within a single 50-to-70-minute class session. This makes conducting realistic in-class activities and demonstrations challenging. Second, students may lack the domain knowledge required to apply Agile methods, resulting in hands-on classroom scenarios that have limited realism. Third, employing Agile methods requires students to assume distinct roles, such as Scrum Master, developer, project stakeholder, and product owner; but opportunities to "act out" key roles can be limited, i.e., not everyone may get a chance to play the Scrum Master. The result of these challenges may be a surface-level, conceptual understanding of Agile concepts that may not capture all the nuances of being part of an Agile project.

There are three broad areas in which Agile methodologies can be taught through an "active learning" approach, with examples for each area from the information systems education literature. There are several ways to accomplish this.

Strongly scoped activities can familiarize students with specific Agile concepts. These activities may focus on some aspect of Agile development (e.g., estimation) or simplified to fit within a classroom time limit. For example, one Scrum activity required students to perform an origami design and development task in three iterations, including a 5-minute development sprint and a 2-minute Scrum meeting [24]. This allowed their entire activity to be completed within 50 minutes. Although possibly sacrificing some degree of realism, this activity addresses both classroom time limits and students' domain knowledge limitations.

Agile as a project requirement is another way to introduce Agile concepts, without the class session time constraints. In this scenario, students are given a group development project where adherence to an Agile methodology is part of the assignment. For example, [25] implemented a nine-week development project in a systems analysis and design course by using students' knowledge from previous courses to keep the focus on applying the Agile methodology. This approach addressed limited student domain knowledge by relying on skills gained from prior courses and "did not require students to learn many new skills" (p. 143). The Scrum Master Role was also rotated among team members each sprint, enabling all students to "act out" this key role.

Integrating agile into the learning process is a third approach to teaching agile concepts. Here, the learning process itself is organized as a series of agile sprints. In their review of the literature, [26] note that IS education increasingly uses "Agile as the method for teaching the content" (p. 275). For

example, [27] had students work in Scrum teams to learn course content, such as case-related activities and exercises, which were organized into a series of sprints. As with [25], the role of Scrum Master was rotated among team members. Their approach also limited the importance of domain knowledge, because agile was the means to acquire knowledge. Sprints spanned multiple days, giving students extended exposure and practice with Agile methods.

4. The Opportunity – Using Novel Technology to Enhance Agile Development

4.1. Agile methodologies in AI and ML projects

The explosion of data and the rise of artificial intelligence (AI) and machine learning (ML) have propelled organizations towards data-driven decision-making. This suggests a shift towards agile methodologies which are known for their flexibility and rapid iteration. Tailoring Agile to analytics and AI, often referred to as *Agile Analytics* (*AA*), can harness the power of Agile while drawing the benefits from data science and AI.

Agile Analytics requires the collaboration of a diverse group of professionals who bring specialized skills to the project, including analysts, data engineers, data scientists, and subject matter experts. Implementing practices such as paired programming, shared code ownership, frequent team discussions, and demonstrations are crucial for developing collective solutions [28]. The AA process prioritizes the iterative formulation and refinement of research questions over the detailed up-front problem identification that is typical in traditional software projects [29]. This approach allows for the scope of inquiry to evolve based on new insights from data, aligning with Agile's flexible and adaptive framework. However, directly applying Agile principles to data-centric initiatives can be ineffective.

Teams operating under Agile principles need autonomy and trust, enabling them to experiment, quickly learn from failures, and share knowledge effectively, avoiding the creation of isolated expertise areas. However, unlike in Agile software development where teams are supposed to be fully self-organizing, the teams in AA typically need to be planned because of the specialized and unique skill sets needed.

In an AA context, data input and processes may change drastically due to the dynamic nature of the business problem, whereas in Agile software development, there is typically more stability around the input and the process (program logic). Therefore, data must be made available to analysts precisely when it is needed to address newly emerging questions. The continuous exploration of new data sources is essential for refining analytical models. In the same vein, data scientists in AA engage in experimental comparisons of modeling techniques, such as evaluating the effectiveness of random forests versus neural networks, to determine the most suitable process for data analysis.

Continuous delivery in AA often involves sharing trained models or the results of data analyses rather than the software products typical of conventional Agile projects. This approach, aimed at providing actionable insights to stakeholders in a timely and iterative manner, exemplifies the Agile commitment to delivering value frequently and incrementally. Early and continuous releases of tools and models, even in their preliminary development stages, encourage stakeholder feedback and can inform subsequent development cycles.

Despite the need for specific adaptations, Agile methodologies offer considerable advantages for managing the uncertainties and empirical nature of data analytics and AI projects. The principles of Agile (e.g., embracing change, facilitating rapid feedback, encouraging collaborative efforts, and enabling flexible planning) can aid organizations in navigating the complexities of analytics and AI initiatives, thereby enhancing the potential to derive meaningful value from these endeavors. Tailoring Agile practices to the distinctive challenges and dynamics of data analytics and AI projects can significantly improve project outcomes.

4.2. The potential impact of Generative AI on Agile methodologies: Reshaping software development practices

The dawn of generative artificial intelligence (AI), especially the emergence of large language models (LLMs) (e.g., *ChatGPT*, *Bard*, *Midjourney*), promises a transformative era for Agile methodologies in software development [30]. These sophisticated AI models possess the remarkable ability to generate extensive textual content from prompts, potentially revolutionizing the creative aspects of Agile practices, such as sprint planning and backlog refinement.

At the heart of Agile lies continuous development and feedback loops, fueled by rapid prototyping and experimentation [31]. Generative AI, with its ability to rapidly generate code snippets and design mockups, is poised to streamline this process. [32] propose an AI-assisted Agile methodology for rapid prototyping, highlighting its effectiveness in uncertain environments. This integration can lead to improved product quality and efficacy by facilitating faster iteration and the exploration of diverse design possibilities.

Generative AI's automation potential extends beyond prototyping. By efficiently generating documentation, user stories, and even basic code snippets, it can free up valuable time for Agile team members [33]. This shift in focus allows them to engage in more complex, strategic tasks, maximizing their expertise and contribution to the project.

Traditionally, requirement analysis can be a time-consuming and subjective process. However, LLMs offer the potential to streamline this process by rapidly generating detailed specifications from minimal input [34]. This not only increases efficiency but can also enhance clarity and alignment with stakeholder expectations, ultimately leading to the delivery of products that more closely align with meet user needs. This opportunity is potent when organizational focused LLMs (i.e., those trained exclusively on organizational data) are deployed.

The introduction of LLMs may empower product owners within Agile environments to extend their abilities to coding and querying of databases. Product owners can potentially reduce their reliance on developers and database administrators, allowing them to exert greater influence over project outcomes [35]. This shift could reshape traditional roles and responsibilities within Agile teams fostering a more collaborative and autonomous work environment.

While the integration of generative AI into Agile processes holds immense promise, it is crucial to acknowledge the associated challenges. Issues like interpretability, potential biases in AI outputs, and the need for careful validation require careful consideration [36]. Agile teams must adopt a balanced approach to ensure that the contributions of generative AI are accurate, reliable, and ethically sound.

The incorporation of generative AI into Agile methodologies presents a significant opportunity to reshape software development practices. By enhancing creativity, streamlining prototyping and experimentation, automating routine tasks, refining requirement analysis, and redefining team member roles, generative AI has the potential to boost the efficiency and effectiveness of Agile teams. Of course, a balanced and responsible approach is necessary to address the associated challenges and ensure that this technology delivers on its transformative promise. However, by carefully considering the ethical implications and fostering responsible development practices, the industry should be able to unlock the full potential of generative AI in the exciting future of Agile software development.

4.3. Conceptual Modeling in Agile development projects

Conceptual modeling emerged in the 1970s as a phase of IS development that involved conceptualization and representation of the problem to be addressed by IS and the domain the IS sought to represent. Since then, conceptual modeling has proven effective for facilitating and promoting mutual understanding, communication, design, and decision making among the different parties in IS development (e.g., developers, subject-matter experts, target users).

However, conceptual modeling has struggled to remain relevant for Agile development (recall the values of working software over comprehensive documentation and responding to change over following a plan enshrined in the Agile Manifesto [4]). Many Agile practitioners routinely forgo formal conceptual modeling [37]. Research on conceptual modeling for Agile remains scarce, potentially due to the perception that it is practically irrelevant [37, 38, 39, 40]. This is due to two key assumptions in traditional conceptual modeling research and pedagogy that appear to be counter to Agile. First, conceptual modeling is assumed to completely capture the domain of an IS project to guide the design phase (e.g., code, database structures, etc.). This assumption clearly misaligns with Agile, where

relevant facts, requirements, and assumptions emerge over time. Second, much of conceptual modeling assumes the specifications are (semi)formal, in that they precisely and unambiguously define the domain's scope and relevant domain concepts. This misaligns with Agile where clarity and consensus emerges over time and may never be completely resolved. Furthermore, Agile is often lean, informal, and increasingly embraces hypothesis testing when design choices are unclear [15].

The misalignments between traditional conceptual modeling and Agile create what we call the A*gile Conceptual-Modeling Paradox*. Both conceptual modeling and Agile seek to improve understanding, communication, design, and decision-making, and both have been shown to be effective at doing so, but they have failed to be used in tandem with one another. As a result, conceptual modeling, which should further enhance Agile methods by facilitating team communication, shared understanding, and process and product documentation, is not being leveraged by the Agile teams. To resolve this paradox, we need to reconsider the traditional conceptual modeling assumptions. These assumptions result from specific ways in which conceptual modeling has been used in the past (e.g., to guide relational database design), and do not fully reflect the potential or the landscape of modern conceptual modeling. The assumptions should be revisited and systematically relaxed, both when teaching conceptual modeling and when conducting conceptual modeling research.

Conceptual modeling can better embrace the uncertainty, evolution, and incomplete nature of domains and the virtues of informal, messy, and casual modeling. Conceptual modeling can support Agile development in many ways. For example, it can be used to facilitate model-driven engineering. Here, modeling is already practiced to a limited extent for rapid prototyping and can be further effective to support cow-code development [41]. These practices can be expanded to general Agile prototyping applications. More generally, conceptual modeling can also be used in Agile to support sense-making, communication, document development choices, guide user interfaces, database design and application functionality. However, the form and function of conceptual modeling probably needs to be updated. Hence, to ensure alignment with Agile values and practices, modeling needs to promote representations and methods that insist on few rules and embrace trial-and error. Such solutions have already been developed, including extremely lean modeling languages and methods, informal modeling, and automated model generation [42, 43].

These approaches can already be considered when teaching Agile. Furthermore, Agile offers an important impetus for further research into these approaches. In particular, the systematic investigations of the benefits and limitations of these approaches for different types of Agile development is an important research opportunity.

Reconsidering conceptual modeling research and pedagogy may make conceptual modeling more relevant for Agile. Conceptual modeling should become more consistent with Agile, while accommodating the uncertain and evolving nature of modern development efforts and domains.

In sum, the increasing complexity, Agile need, and implications for how to expand Agile Development are all in need of consideration for future development and application.

5. Conclusion and Future Outlook

This paper has proposed an *Agile Development Framework* to serve as a reference to understand Agile development based on its original intent or promise, the reality of its use, and the opportunity for its continued expansion, given increasingly complex applications, new challenges, and the need to guide future work in this important area. Agile development has long been useful for addressing systems development challenges. This paper has examined the past and current uses of Agile, leading to suggestions for how it might progress and be applied in the future. The past includes the origins and notion of Agile as a response to existing, structured methods. The reality of Agile is wide-spread adoption while recognizing Agile-related challenges. With emerging technologies and increasingly important real-world and global problems, there are opportunities to advance Agile for a wide range of development topics. Finally, since Agile has become widely adopted in practice, educational considerations for the next generation of the workforce that will use Agile is important to ensure its continued adoption in practice.

The Framework proposed in this paper has been only partially explored. Although some aspects may not have been fully articulated above, they all deserve further development and deeper examination beyond what has been presented here. Further, it may be necessary to modify or expand the framework

itself. However, we have identified some significant implications for future research on the types of applications to which Agile can be applied, the increasing complexity of the problem applications, and the growing need to expand and apply Agile.

6. References

[1] Rigby, D.K., Sutherland, J., & Takeuchi, H. 2016. Embracing Agile: How to master the process that's transforming management, *Harvard Business Review*, May, 40-50.

[2] Vidgen, R., & Wang, X. 2009. Coevolving systems and the organization of Agile software development, *Information Systems Research*, 20(3), 355-376.

[3] Schmidt, C., Kude, T., Tripp, J., Heinzl, A., & Spohrer, K. 2013. Team adaptability in Agile information systems development, International Conference on Information Systems (ICIS), Milan, Italy.

[4] Beck, A., Grenning, J., Martin, R. C., Beedle, M., Highsmith, J., Mellor, S. van Bennekum, A., Hunt, A. Schwaber, K. Cockburn, A. Jeffries, R., Sutherland, J., Cunningham, W. Kern, J., Thomas, D., Fowler, M. Marick, B. 2001. Manifesto for agile software development. Agile Alliance. Retrieved from http://agilemanifesto.org/. (Accessed August 3, 2019).

[5] Jason, L.A., Stevens, E., Ram, D. Miller, S.A., Beasley, C.A., & Gleason, K.D. 2016. Theories in the field of community practice, *Global Journal of Community Psychology Practice*, 7(2), 1-27.

[6] Tripp, J.F. & Armstrong, J.F. 2016. Agile methodologies: Organizational adoption motives, tailoring, and performance, *Journal of Computer Information Systems*, 10 pgs. DOI: 10.1080/08874417.2016.1220240. Turetken, O., Stojanov, I., & Trienekens, J. J. (2017). Assessing the adoption level of scaled Agile development: A maturity model for Scaled Agile Framework, *Journal of Software: Evolution and Process*, *29*(6), e1796.

[7] Boehm, B. 1988. A Spiral Model of software development and enhancement, *Computer*, May, 61-72.

[8] Larman, C. & Basili, V.R. 2003. Iterative and Incremental Development: A Brief History, *Computer*, June, 2-11.

[9] Sacolick, I. 2022. A brief history of the Agile methodology. URL: https://www.infoworld .com/article/3655646/a-brief-history-of-the-agile-methodology.html (Accessed April 26, 2024).

[10] Peters, L.J. & Tripp, L.L. 1976. Is software design "wicked"?, *Datamation*, 22, 5, May, 127-136.

[11] Rigby, D., Elk, S., & Berez, S. 2020. *Doing Agile Right: Transformation Without Chaos*. Boston, MA: Harvard Business Review Press.

[12] Sutherland, J. & Sutherland, J.J. 2014. *SCRUM: The Art of Doing Twice the Work in Half the Time*, New York, NY: Currency.

[13] Costa, D. I. C., Filho, E. P. S., Da Silva, R. F., De C. Quaresma Gama, T. D., & Cortés, M. I. 2020. Microservice architecture: A tertiary study, ACM International Conference Proceeding Series.

[14] Zhou, X., Li, S., Cao, L., Zhang, H., Jia, Z., Zhong, C., Shan, Z., & Babar, M. A. (2023). Revisiting the practices and pains of microservice architecture in reality: An industrial inquiry. *Journal of Systems and Software*, *195*, 111521. URL: https://doi.org/10.1016/j.jss.2022.111521.

[15] Schwartz, M. 2017. *A Seat at the Table and The Art of Business Value*. IT Revolution.

[16] Fitzgerald, B., & Stol, K.-J. 2017. Continuous software engineering: A roadmap and agenda, *Journal of Systems and Software*, *123*, 176–189.

[17] Hemon-Hildgen, A., Rowe, F., & Monnier-Senicourt, L. 2020. Orchestrating automation and sharing in DevOps teams: A revelatory case of job satisfaction factors, risk, and work conditions, European journal of Information Systems, 29(5), 474-499.

[18] Turetken, O., Stojanov, I., & Trienekens, J.J.M. 2017. Assessing the adoption level of scaled agile development: A maturity model for Scaled Agile Framework, *Journal of Software Evolution and Process*, 29, e1796. DOI: 10.1002/smr.1796.

[19] Fuchs, C., and Hess, T. 2018. Becoming Agile in the digital transformation: The process of a large-scale Agile transformation, *International Conference on Information Systems*, San Francisco, USA.

[20] Demirkan, H., & Spohrer, J. 2015. T-shaped innovators: Identifying the right talent to support service innovation, *Research-Technology Management, 58*(5), 12-15.
[21] Fitriani, W.R., Rahayu, P., & Sensuse, D.I. 2016. Challenges in Agile software development: A systematic literature review." ICACSSIS 2016, 155-164. URL: http://ieeexplore.ieee.org/ document/7872736/ (Accessed April 26, 2024).
[22] Bick, S., Spohrer, K., Hoda, R., Scheerer, A., & Heinzl, A. 2017. Coordination challenges in large-scale software development: A case study of planning misalignment in hybrid settings, *IEEE Transactions on Software Engineering*, 44(10), 932-950. https://doi.org/10.1109/ TSE.2017.2730870.
[23] Poza-Vilches, F., et al. 2019. A systematic review of the use of Agile methodologies in education to foster sustainability competencies, *Sustainability*, 11(10), 2915-2929. URL: https://doi.org/10.3390/su11102915.
[24] Sibona, C., Pourreza, S., & Hill, S. 2018. Origami: An active learning exercise for Scrum project management, *Journal of Information Systems Education*, 29(2), 105-116.
[25] Baham, C. 2019. Teaching Tip: Implementing Scrum wholesale in the classroom, *Journal of Information Systems Education*, 30(3), 141-159.
[26] Sharp, J. H., Mitchell, A., & Lang, G. 2020. Agile teaching and learning in information systems education: An analysis and categorization of literature, *Journal of Information Systems Education*, 31(4), 269-281.
[27] Rush, D. E. & Connolly, A. J. 2020. An Agile framework for teaching with Scrum in the IT project management classroom, *Journal of Information Systems Education*, 31(3), 196-207.
[28] Saltz, J., & Shamshurin, I. 2016. Big data team process methodologies: A literature review and the identification of key factors for a project's success. 2016 IEEE International Conference on Big Data (Big Data). URL: https://doi.org/10.1109/BIGDATA.2016.7840719.
[29] Cao, L., & Ramesh, B. 2008. Agile requirements engineering practices: An empirical study, IEEE Software, 25(1), 60-67. https://doi.org/10.1109/MS.2008.1.
[30] Brown, T. B., Mann, B., Ryder, E., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei,D. 2020. Language models are few-shot learners. DOI: 10.48550/arXiv.2005.14165.
[31] Beck, K. 2002. *Agile Software Development: Principles, Patterns, and Practices*. Boston, MA: Addison-Wesley Professional.
[32] Cai, W., Zhang, Y., Xu, Z., Zou, W., & Zhou, M. 2021. An AI-assisted agile methodology for rapid prototyping in uncertain environments, *Information Sciences*, 569, 155-172.
[33] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.& Polosukhin, I. 2017. Attention is all you need, DOI: 10.48550arXiv.1706.03762.
[34] Sharma, N., Khandelwal, K., & Gupta, T. 2022. A review of natural language processing-based approaches for requirements engineering in agile methodologies, *International Journal of Information Technology*, 14(6), 3492-3506.
[35] Deloitte Insights 2023. Harnessing the power of generative AI: An Agile methodology for uncertain times. URL: https://www2.deloitte.com/nl/nl/pages/risk/articles/the-power-of-generative-ai-an-agile-methodology.html (This link has been removed as it was not a trustworthy source).
[36] European Commission. 2019. Ethics guidelines for trustworthy AI. URL: https://commission. europa.eu/ index_en.
[37] Michael, J., Bork, D., Wimmer, M., & Mayr, H. C. 2024. Quo vadis modeling? Findings of a community survey, an ad-hoc bibliometric analysis, and expert interviews on data, process, and software modeling." *Software and Systems Modeling*, *23*(1), 7-28.
[38] Recker, J.C., Lukyanenko, R, Jabbari, M., Samuel, B., & Castellanos, A. 2021. From representation to mediation: A new agenda for conceptual modeling research in a digital world." *MIS Quarterly*, 45(1) 269-300.
[39] Lukyanenko, R., Storey, V., & Pastor, O. 2022. System: A core conceptual modeling construct for capturing complexity, *Data & Knowledge Engineering*, 141, 102062.
[40] Storey, V.C., Lukyanenko, R., & Castellanos, A. 2023. Conceptual modeling: Topics, themes, and technology trends, *ACM Computing Surveys*, 55(14s), 1-38.
[41] Lano, K., Kolahdouz-Rahimi, S., Troya, J., & Alfraihi, H. 2022. Introduction to the theme section on Agile model-driven engineering, *Software and Systems Modeling*, *21*(4), 1465-1467.

[42] Castellanos, A., Tremblay, M. C., Lukyanenko, R., & Samuel, B. 2020. Basic classes in conceptual modeling: theory and practical guidelines, *Journal of the Association for Information Systems*, *21*(4), 3.
[43] Lukyanenko, R., Samuel, B. M., Parsons, J., Storey, V. C., & Pastor, O. 2023. Datish: A universal conceptual modeling language to model anything by anyone, *ER Forum 2023*, 1–14.