

Emerging Trends in Software Engineering: A Comprehensive Review

Ibrahim M.I, Ibrahim M, Ibrahim A, Nura T.A, Maigari A and Umar M

Department of Mobile and Satellite Communication, Jigawa State Institute of Information Technology, Kazaure, Nigeria

Corresponding Author*

Ibrahim M.I
Department of Mobile and Satellite Communication,
Jigawa State Institute of Information Technology,
Kazaure, Nigeria
E-mail: ibrahimq7@jsiit.edu.ng

Copyright: © 2024 Ibrahim M.I, et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Received: October 05, 2023, Manuscript No. IJIRSET-23-115856;
Editor assigned: October 07, 2023, PreQC No. IJIRSET-23-115856 (PQ); **Reviewed:** October 21, 2023, QC No. IJIRSET-23-115856;
Revised: January 03, 2024, Manuscript No. IJIRSET-23-115856 (R);
Published: January 10, 2024, DOI: 10.35248/IJIRSET.24.05(1).001

Abstract

Software engineering is a dynamic field that constantly evolves to meet the ever-changing demands of technology and industry. This research paper aims to provide a comprehensive review of the emerging trends in software engineering, shedding light on the latest developments, methodologies, and technologies shaping the industry. The paper explores various key areas of software engineering, including agile methodologies, DevOps, Artificial Intelligence (AI) and Machine Learning (ML), cloud computing. By examining these trends, their impact on software development, how they differed from traditional approaches, and their potential future implications, this research paper offers valuable insights into the evolving landscape of software engineering.

Keywords: Agile • DSDM • Traditional • DevOps • Cloud • AI/ML

Introduction

As technology continues to advance rapidly, the field of software engineering is constantly evolving to meet new challenges and opportunities. Emerging software engineering trends represent the forefront of innovation, shaping the way software is developed, deployed, and utilized. Some of the key emerging trends in software engineering, include artificial intelligence and machine learning, microservices architecture, low-code/no-code development, DevOps and Continuous Integration/Continuous Deployment (CI/CD), server less computing, Internet of Things (IoT), software development, blockchain and distributed ledger technologies and edge computing among others [1].

These emerging software engineering trends reflect the industry's ongoing quest for innovation, efficiency, and improved user experiences. So, software engineers must adapt to these trends to stay ahead in an ever-changing technological landscape.

Objectives

- To help readers understand the concept of software engineering and emerging trends in the field.
- To help developers and practitioners understand the benefits of agile development approach over other traditional methods.
- To compare agile methodologies with traditional approaches in terms of project outcomes, adaptability to change as well as team dynamics.
- To examine how technology trends impact software development processes.

Background

Software engineering is a crucial field that addresses the design, development, testing, deployment, and maintenance of software systems. It emerged as a response to the growing complexity of software applications and the need for systematic approaches to building and managing large-scale software projects. The background and significance of software engineering can be understood by the growing software complexity and crisis [2].

In the 1960's, computer scientists and software developers began to face what was known as the "software crisis." This crisis was characterized by the inability to meet project deadlines, budget overruns, and a high rate of software project failures. The need for a systematic, engineering-based approach to software development was recognized [3].

So, in response to the software crisis, the term "software engineering" was coined at the NATO Science Committee's 1968 Garmisch Conference. The goal was to apply engineering principles to software development, treating it as a disciplined and repeatable process.

Literature Review

Agile software development methodologies

Agile methodologies are a set of principles and practices designed to promote flexibility, collaboration, and adaptability in software development and project management.

The agile approach contrasts with traditional, linear project management methods (like the waterfall model) that follow a sequential and rigid process. Instead, agile methodologies emphasize iterative development, frequent feedback, and continuous improvement. Some popular agile methodologies include the following [4]:

Extreme Programming (XP): XP emphasizes technical practices to ensure high-quality software, such as Test-Driven Development (TDD), continuous integration, pair programming, and frequent releases. It aims to deliver valuable software quickly while maintaining a high level of quality.

Lean software development: Inspired by Lean manufacturing principles, Lean software development focuses on minimizing waste, maximizing value, and optimizing the whole development process. It encourages early and frequent deliveries, empowering teams to adapt to changes rapidly.

Scrum: Scrum is a widely-used agile framework that organizes work into fixed-length iterations called "sprints," usually lasting two to four weeks. It employs a set of predefined roles (Scrum master, product owner, and development team) and ceremonies (Daily standups, sprint planning, sprint review, and sprint retrospective) to facilitate collaboration and incremental development.

Kanban: Kanban is based on the visualization of work items on a Kanban board, allowing teams to manage their workflow more efficiently. It promotes a continuous flow of work, with tasks moving from "To Do" to "In Progress" to "Done" columns, focusing on limiting work in progress and promoting a steady delivery pace.

We should note that, the adoption of agile methodologies has led to numerous benefits, including improved customer satisfaction, reduced project risks, better adaptability to changing requirements, and increased collaboration within development teams. However, it also requires commitment and active participation from all team members and stakeholders to be truly effective.

Evolution of agile methodologies

The evolution of agile methodologies is a fascinating journey that reflects the software development industry's continuous search for better ways to deliver high-quality products efficiently. The agile movement originated in the 1990's, and over time, it has evolved and diversified into various methodologies and frameworks. Here's a brief overview of the key milestones in the evolution of agile methodologies:

Incremental and iterative development (1970's-1980's): The roots of agile can be traced back to the incremental and iterative development practices introduced in the 1970's and 1980's. Early software developers began to realize the benefits of breaking large projects into smaller, manageable increments and iterating on them to improve the product continuously.

Adaptive Software Development (ASD) and crystal (1990's): In the early 1990's, Jim Highsmith developed adaptive software development, which emphasized collaboration, incremental development, and embracing change. Around the same time, Alistair Cockburn introduced crystal methodologies, which focused on tailoring the development process based on the project's unique characteristics and team size.

Scrum (Mid 1990's): Scrum, one of the most popular agile frameworks, was formalized in the mid 1990's by Jeff Sutherland and Ken Schwaber. It draws inspiration from Lean manufacturing and emphasizes short, time-boxed iterations (sprints), cross-functional teams, and frequent inspection and adaptation through ceremonies like daily standups, sprint planning, and retrospectives.

Extreme Programming (XP) (Late 1990's): XP, developed by Kent Beck and others, emerged in the late 1990's. It promoted engineering practices such as Test-Driven Development (TDD), pair programming, continuous integration, and small, frequent releases. XP aimed to deliver high-quality software in a flexible and customer-centric manner.

Agile manifesto (2001): The term "Agile" gained prominence in 2001 when seventeen software development practitioners, including representatives from Scrum, XP, ASD, and other approaches, met to define the core values and principles of agile software development. This resulted in the agile manifesto, which laid the foundation for the agile movement.

Agile methodologies proliferation (2000's): Throughout the 2000's, various agile methodologies and frameworks gained popularity, such as Feature-Driven Development (FDD), Dynamic Systems Development Method (DSDM), and Kanban. Each methodology offered its unique approach to agile software development, catering to diverse project requirements.

Scaling agile (2010's): As agile adoption spread to larger organizations and complex projects, the need for scaling agile methodologies arose. Frameworks like Scaled Agile Framework (SAFe), Large-Scale Scrum (LeSS), and Disciplined Agile Delivery (DAD) were developed to address the challenges of scaling agile practices across multiple teams and departments.

Modern trends and customization (Present): In recent years, agile has continued to evolve and adapt to changing industry needs and technological advancements. Trends like DevOps, Continuous Integration/Continuous Deployment (CI/CD), and Lean agile have further influenced agile methodologies' evolution, promoting continuous delivery and increased collaboration between development and operations teams.

Throughout its evolution, agile methodologies have sparked a paradigm shift in software development, promoting customer collaboration, team empowerment, and a focus on delivering valuable software iteratively. Agile remains a dominant force in modern software development, and its principles continue to shape how teams approach projects and respond to ever-changing business needs.

Agile manifesto and principles

The agile manifesto, created in 2001 by a group of seventeen software development practitioners, is a foundational document that defines the core values and principles of agile software development. It emphasizes a flexible and customer-centric approach to delivering software, focusing on collaboration, adaptability, and delivering value [5].

Below are the four (4) key values of the agile manifesto that software developers used to guide their development tasks.

Agile manifesto values

- **Individuals and interactions over processes and tools:** Agile values the importance of people working together effectively over relying solely on rigid processes or tools. Good communication and collaboration are essential to the success of agile teams.
- **Working software over comprehensive documentation:** While documentation has its place, agile prioritizes delivering functional software that provides value to customers. Working software is the primary measure of progress.
- **Customer collaboration over contract negotiation:** Agile fosters close collaboration with customers and stakeholders throughout the development process. Customer feedback is actively sought and used to ensure the delivered software meets their needs.
- **Responding to change over following a plan:** Agile recognizes that requirements and priorities can change during a project's course. Instead of rigidly adhering to initial plans, agile teams embrace change and adapt their approach to maximize value.

Similarly, the twelve (12) principles outlined in the agile manifesto:

Agile manifesto principles

- **Satisfy the customer through early and continuous delivery of valuable software:** Agile teams aim to deliver functional software in short iterations, providing customers with valuable features early and frequently. Early delivery of deliverables does not mean quality is compromised.
- **Welcoming changing requirements, even late in development:** Agile projects expect and accommodate changes in requirements, embracing them as opportunities to improve the software's fit with customer needs. Customer sophistication in terms of requirement changes is one of the main reasons why agility was given birth to. To deny customers of requirement changes in the course of the development and delivery process is like going back to the traditional methods of software development, and not agile methodology.
- **Deliver working software frequently, with a preference for shorter timescales:** Frequent delivery of working software allows for faster feedback and enables teams to adjust their approach based on real-world results.
- **Collaborate daily between business people and developers throughout the project:** Regular collaboration between the business side and development teams ensures a shared understanding of project goals and requirements.
- **Build projects around motivated individuals, giving them the support and trust they need to get the job done:** Empowered and motivated individuals are more likely to produce high-quality results. Agile teams trust their members to make decisions and value their expertise.
- **Use face-to-face communication whenever possible:** Face-to-face communication is highly valued in agile, as it tends to foster better understanding, improve collaboration, and build stronger relationships between team members.

- **Working software is the primary measure of progress:** Rather than relying on comprehensive documentation or completed tasks, agile measures progress based on the delivery of functional, tested software.
- **Maintain a sustainable pace of work for the development team:** Agile teams avoid burnout and maintain productivity by working at a sustainable pace over the long term. Agility requires constant pace so that projects can be delivered within the shortest possible time. This is key, and without it, there is nothing like agility.
- **Continuous attention to technical excellence and good design enhances agility:** Technical practices such as refactoring, automated testing, and continuous integration are essential to maintaining code quality and enabling rapid responses to change.
- **Simplicity—the art of maximizing the amount of work not done is essential:** Agile teams aim to minimize unnecessary work and complexity, focusing on delivering the most valuable features.
- **Self-organizing teams make the best architectures, requirements, and designs emerge:** Empowered teams that collaborate effectively often produce better solutions than those dictated by external authorities.
- **At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly:** Agile teams engage in regular retrospectives to review their processes, identify areas for improvement, and make the necessary adjustments to enhance their performance.
- **Agile in regulated industries:** Industries with strict regulatory requirements, such as healthcare and finance, are increasingly adopting agile practices. Companies are finding ways to balance agile's flexibility with the need for compliance and documentation.
- **Data-driven agile:** Teams are leveraging data and metrics to make informed decisions, monitor progress, and continuously improve their agile processes. Data-driven agile helps identify areas for improvement and validates the impact of changes.
- **DevOps and continuous delivery:** The integration of DevOps practices with agile methodologies continues to gain prominence. Automating deployment pipelines and focusing on continuous delivery enable faster and more reliable software releases.
- **Agile coaching and mentoring:** Agile coaching has become essential to support teams and organizations in their agile journey. Agile coaches and mentors help teams understand and apply agile principles effectively.
- **Focus on well-being and psychological safety:** Agile organizations are recognizing the importance of psychological safety and well-being in fostering a healthy and productive work environment. Emphasizing team health and work-life balance is becoming a priority.

Impact of agile on software development

The impact of agile on software development has been profound and far-reaching. Agile methodologies have revolutionized the way software is planned, developed, and delivered, leading to numerous benefits for both development teams and stakeholders. Some of the key impacts of agile on software development include [7]:

- **Customer-centric approach:** Agile places a strong emphasis on customer collaboration and delivering value to end-users. By involving customers throughout the development process and seeking frequent feedback, agile ensures that the software aligns with user needs and expectations.
- **Iterative and incremental development:** Agile practices promote iterative development and continuous delivery of small, functional increments of software. This approach allows for early and frequent releases, enabling faster time-to-market and faster response to changing requirements.
- **Improved quality and flexibility:** Agile methodologies, with their focus on continuous testing and feedback, help identify and address issues early in the development process. This leads to higher software quality and the ability to adapt to changes efficiently.
- **Empowered and engaged teams:** Agile principles empower development teams to make decisions, collaborate effectively, and take ownership of their work. This autonomy fosters a sense of ownership and motivation among team members.
- **Transparency and visibility:** Agile practices encourage transparent communication, visible progress tracking, and open access to project information. This enhances trust between stakeholders and the development team, leading to better decision-making.
- **Rapid adaptation to changes:** Agile's flexibility and ability to embrace change allow software development projects to adapt quickly to evolving requirements, market conditions, and technological advancements.
- **Early risk identification and mitigation:** The iterative nature of agile development enables early identification of potential risks and issues, allowing teams to address them proactively before they escalate.
- **Continuous improvement:** Agile methodologies promote a culture of continuous improvement. Regular retrospectives and feedback loops encourage teams to analyze their processes and make necessary adjustments to enhance productivity and efficiency.

These four (4) values and twelve (12) principles serve as the guiding philosophy for agile methodologies and encourage teams to embrace flexibility, adaptability, and continuous improvement in their software development processes.

Discussion

Modern trends in agile practices

We need to remember that trends in agile practices can vary depending on the industry, organizational culture, and technological advancements. It's essential for teams and organizations to stay informed, remain open to experimentation, and continuously adapt their agile practices to achieve the best results [6].

- **Remote and distributed agile:** With the increasing adoption of remote work and distributed teams, agile practices have adapted to facilitate collaboration and communication across geographical locations. Virtual agile ceremonies, collaborative online tools, and video conferencing have become more prevalent to maintain team cohesion.
- **Agile at scale:** Large organizations are embracing agile methodologies at the enterprise level to coordinate multiple teams working on complex projects. Frameworks like Scaled Agile Framework (SAFe), Large-Scale Scrum (LeSS), and Disciplined Agile Delivery (DAD) provide guidance on how to scale agile practices effectively.
- **Value stream mapping:** Teams are focusing on understanding the entire value stream from idea to delivery, identifying bottlenecks and inefficiencies. Value stream mapping helps optimize workflows and reduce lead times, improving overall delivery speed and customer value.
- **Agile for non-software projects:** Agile principles are expanding beyond software development to various other industries and domains, such as marketing, human resources, and product management. The application of agile concepts outside of IT is gaining traction.
- **Design thinking and agile:** Combining design thinking principles with agile practices has become popular to emphasize a strong customer focus and deliver products that truly meet user needs. The iterative nature of both approaches complements each other well.

- **Reduced waste:** Agile encourages a focus on delivering the most valuable features and eliminating unnecessary work, leading to reduced waste of resources and effort.
- **Increased customer satisfaction:** The close collaboration with customers and frequent deliveries of working software result in higher customer satisfaction and better alignment of the final product with customer expectations.
- **Shorter time-to-market:** Agile's iterative approach allows for early delivery of valuable features, which can lead to a shorter time-to-market for software products.
- **Enhanced team communication and collaboration:** Agile ceremonies like daily standups, sprint reviews, and retrospectives facilitate regular communication and collaboration among team members, fostering a shared understanding of goals and progress.

Generally, agile has transformed software development by promoting a culture of adaptability, customer-centricity, collaboration, and continuous improvement. The impact of agile extends beyond development teams, positively influencing project success rates, customer satisfaction, and the ability to deliver software that meets users' needs effectively.

Cloud computing

Cloud computing is a transformative technology that has revolutionized the way businesses and individuals access and use computing resources. It refers to the delivery of various computing services, including servers, storage, databases, networking, software, and analytics, over the internet (the cloud). Instead of maintaining physical servers and data centers on-site, users can access and utilize these resources on-demand from cloud service providers [8].

Impact of cloud computing on software engineering

Cloud computing has had a profound impact on software engineering, transforming the way software is developed, deployed, and maintained. It has become an integral part of the software development lifecycle, offering numerous benefits and new possibilities for developers and businesses.

Besides, cloud computing has revolutionized software engineering by providing scalable, cost-effective, and innovative solutions. It has empowered developers to build, deploy, and manage applications more efficiently, leading to improved software quality, faster delivery cycles, and increased business agility. As cloud technology continues to evolve, its impact on software engineering is expected to grow, driving further advancements in application development and deployment practices.

Potential future trends in software engineering

As software engineering continues to evolve, several potential future trends are expected to shape the industry. These trends are driven by advancements in technology, changes in user expectations, and the need to address emerging challenges [9].

These potential future trends in software engineering are expected to shape the industry, drive innovation, and create new opportunities for developers, businesses, and end-users alike. Embracing these trends and staying updated on emerging technologies will be crucial for software engineers to remain competitive in the evolving landscape.

Ethical considerations and challenges

Ethical considerations and challenges are critical aspects of software engineering, especially as technology continues to advance and impact various aspects of people's lives. The decisions made by software engineers can have far-reaching consequences, both positive and negative.

Attempt to address these ethical considerations and challenges requires a multi-stakeholder approach involving software engineers, policymakers, researchers, and users. Ethical guidelines, codes of conduct, and industry standards can help guide software engineers in making ethical decisions throughout the software development lifecycle. Regular reflection and dialogue on ethical issues will be vital to ensure that technology continues to benefit society while minimizing its negative impacts.

Skill sets and workforce requirements

In the ever-evolving field of software engineering, there is a wide range of skill sets and workforce requirements that are essential for success. Software engineers and developers need to possess technical expertise, as well as soft skills to thrive in collaborative and dynamic work environments.

Therefore, software engineering as a discipline requires a combination of technical expertise, soft skills, and an adaptive mindset. The ability to learn, collaborate, and adapt to changing technology trends is vital for success in this dynamic field. As software becomes more intertwined with various industries and domains, the demand for skilled software engineers continues to grow, making software engineering a promising and rewarding career path.

Conclusion

In conclusion, software engineering is a dynamic and rapidly evolving field that continues to shape the way we live, work, and interact with technology. From the agile methodologies that promote flexibility and collaboration to the groundbreaking impact of artificial intelligence and machine learning, software engineering has undergone significant transformations.

The agile manifesto, with its principles of customer collaboration, iterative development, and responding to change, has revolutionized software development practices, enabling teams to deliver high-quality software with greater efficiency and responsiveness to user needs.

DevOps, as a cultural and technical movement, has further bridged the gap between development and operations teams, promoting continuous integration, continuous delivery, and a focus on automation to streamline the software development lifecycle. While, blockchain technology has brought about secure and transparent decentralized solutions, paving the way for novel applications in various industries beyond crypto currencies.

Artificial Intelligence (AI) and Machine Learning (ML) are transforming software engineering by automating tasks, improving decision-making processes, and enabling the development of sophisticated applications that learn and adapt. Cloud computing is another trend that revolutionized how software is deployed, scaled, and accessed, providing businesses with flexible and cost-effective solutions to meet their computing needs.

The rise of microservices and containerization has enabled greater flexibility, scalability, and resilience in software development, breaking down complex applications into smaller, manageable components.

While these emerging trends present immense opportunities, they also come with their share of challenges, such as data privacy, security concerns, ethical considerations, and the need for continuous learning and adaptation.

In this dynamic landscape, software engineers must possess a diverse skill set that includes technical expertise, communication skills, adaptability, and a strong ethical foundation. By embracing a user-centric approach, collaboration, and innovation, software engineering teams can address the challenges and harness the potential of emerging trends to create software solutions that positively impact society.

As software engineering continues to evolve, the pursuit of excellence, ethical practices, and a commitment to lifelong learning will be essential to driving innovation and shaping a brighter future for the world of technology.

Recommendations

Further research in software engineering can explore various areas to address emerging challenges, enhance existing practices, and push the boundaries of innovation. Such as ethical issues in Artificial Intelligence (AI) and Machine Learning (ML), for instance, investigation of ethical considerations and bias mitigation techniques in AI and ML algorithms to ensure fairness, transparency, and accountability in automated decision-making systems. And also AI-driven approaches to automated software testing to improve test coverage, detect defects, and optimize testing processes.

References

1. Lakhani, A. "The Role of DevOps in Digital Transformation." *Peerbits*. (2023).
2. Albayrak, O., et al. "Incomplete software requirements and assumptions made by software engineers." 2009 16th Asia-Pacific Software Engineering Conference, Batu Ferringhi, Malaysia. *IEEE*. (2009).
3. Dey, P.P., et al. "Best practices for improving user interface design." *Int J Soft Eng Appl*. 10.5 (2019): 71-83.
4. Sommerville, I. "Agile software development." *Soft Eng*. (2015): 72-98.
5. Beck, K., et al. "What is the Agile Manifesto?" Agile Alliance Group, USA (2019).
6. Indira Gandhi National Open University. "Emerging trends in software engineering." I.G.N.O.U., *Sch Comp Inform Sci*. (2021): 1-19.
7. Koi-Akrofi, G.Y., et al. "IT project success: Practical frameworks based on key project control variables." *Int J Soft Eng Appl*. 10.5 (2019): 55-69.
8. Koi-Akrofi, G.Y., et al. "Understanding the characteristics, benefits and challenges of agile it project management: A literature based perspective." *Int J Soft Eng Appl*. 10.5 (2019): 25-44.
9. Zaitsev, A., et al. "Reviewing the role of the agile manifesto and agile methods in literature." Twenty-fourth Americas Conference on Information Systems, New Orleans, USA (2018).