Shift Left: Moving the Inclusion of Accessibility Functionalities to the Left in Agile Product Development Life Cycle

Sunil Kumar Suvvari

Independent Researcher, Agile Project Management Consultant, USA, Email: yproductsense@gmail.com

Received: 18.10.2023 Revised: 20.11.2023 Accepted: 15.12.2023

ABSTRACT
In modern software development, ensuring accessibility is a critical aspect of creating inclusive digital products. Traditionally, accessibility has been addressed late in the development process, often leading to costly rework and poor user experiences for individuals with disabilities. The "Shift Left" approach advocates for moving accessibility considerations earlier in the Agile product development life cycle, embedding accessibility from the initial stages of requirements gathering, design, and development. This paper explores how shifting accessibility to the left can lead to more efficient workflows, higher-quality products, and better compliance with regulations such as WCAG and ADA. By integrating accessibility testing and principles throughout the Agile process, teams can reduce development costs, enhance product usability, and foster an inclusive design culture. The paper also discusses challenges, benefits, and practical steps for adopting this approach, supported by diagrams to illustrate key concepts.

Keywords: Accessibility, Shift Left, Agile Development, Inclusive Design

1. INTRODUCTION

In today's fast-paced digital environment, creating inclusive products that cater to all users, including those with disabilities, has become imperative. Accessibility is no longer an afterthought but a crucial aspect of software design and development. However, in many agile product development cycles, accessibility considerations are still addressed late in the process, leading to inefficiencies, higher costs, and often, suboptimal results. The "shift left" philosophy, which emphasizes addressing issues and incorporating essential features early in the development cycle, can improve accessibility outcomes by embedding these considerations from the outset.

This paper aims to discuss the role of the "shift left" approach in moving accessibility functionalities to the forefront of agile product development. It also explores the intersection of this shift with security and compliance, drawing from Suvvari (2024), and the concept of the architectural runway in agile methodologies, as elaborated by Suvvari (2024).

1.1. Accessibility in Software Development

In today's increasingly digital world, ensuring accessibility in software is no longer optional. Accessibility refers to the design and development of products, devices, services, or environments for people with disabilities. By adhering to accessibility standards, developers ensure that people with diverse abilities, such as those with visual, auditory, cognitive, or physical impairments, can effectively use digital products(Bevan et. al., 2015).

Governments and organizations around the world have set guidelines to make digital content accessible. Standards like the Web Content Accessibility Guidelines (WCAG) and regulations such as the Americans with Disabilities Act (ADA) and Section 508 mandate compliance for public-facing digital services. However, many software development teams still treat accessibility as an afterthought, addressing it late in the software development life cycle (SDLC). This often results in significant rework, compliance issues, and an unsatisfactory user experience for individuals with disabilities.

1.2. Shift Left Approach

The Shift Left concept in software development advocates moving key activities like testing, security, and now accessibility, to earlier phases of the SDLC. By addressing these critical concerns upfront, teams can detect and resolve issues when they are cheaper and easier to fix, rather than waiting until later stages, where the cost of fixing defects grows exponentially.

Incorporating accessibility early in the development process has traditionally been overlooked, resulting in products that are inaccessible or require significant changes during the testing phase or post-release(Black & Harrison, 2019). By shifting accessibility left, teams proactively ensure that products are designed with inclusivity in mind from the very beginning, aligning accessibility with other core software qualities such as security and performance.

1.3. Purpose of the Paper

The primary purpose of this paper is to:

- Highlight the importance of accessibility in modern software products and its relevance in today's digital-first world.
- Explain the Shift Left approach to embedding accessibility into the Agile product development life cycle.
- Present actionable strategies for integrating accessibility early in the development process, across various Agile phases such as requirements gathering, design, development, and testing.
- Illustrate the challenges and benefits of moving accessibility to the left in the development cycle, supported by diagrams that visualize the integration points and potential impact on overall product quality.

By adopting this approach, development teams can build more inclusive products while simultaneously reducing the time and costs associated with retrofitting accessibility at later stages of the SDLC.

2. Agile Product Development Life Cycle

Agile is a widely adopted software development methodology that emphasizes flexibility, collaboration, and rapid iteration. Unlike traditional models such as Waterfall, where development follows a linear path from planning to deployment, Agile embraces an incremental approach. Teams develop software in small, iterative cycles called sprints, where they continuously gather feedback and adapt to changing requirements. This makes Agile well-suited for incorporating evolving needs like accessibility(Brown, 2021).

The Agile development life cycle typically includes the following phases:

2.1. Requirements Gathering

In Agile, requirements are not fully defined upfront. Instead, they evolve through close collaboration between development teams, product owners, and stakeholders. User stories, which describe product features from an end-user perspective, are created to capture these evolving requirements(Clark & DiFilippo, 2018). Agile teams rely on this flexible approach to iteratively prioritize features based on business needs.

Key Elements:

- User Stories: Feature descriptions written in simple, non-technical language.
- Backlog Prioritization: A dynamic list of features, prioritized based on business value and stakeholder feedback.
- Stakeholder Collaboration: Continuous interaction with business users to refine requirements.

2.2. Design

The design phase focuses on creating visual layouts and user experiences. Agile encourages a flexible approach to design, promoting continuous collaboration between designers, developers, and testers. In this phase, designers build wireframes, mockups, and prototypes to validate ideas quickly and gather feedback before moving into development.

Key Elements:

- Wireframes: Simple sketches outlining the structure and flow of the application.
- Prototypes: Interactive versions of the product that allow for early user testing.
- Design Reviews: Frequent feedback loops to ensure design decisions align with user needs.

2.3. Development

During development, Agile teams follow the sprint cycle, typically lasting two to four weeks. In each sprint, developers focus on implementing a specific set of features from the product backlog. Agile's iterative nature encourages continuous improvement and early feedback.

Key Elements:
- Sprint Planning: A meeting where teams commit to delivering a specific set of user stories within the sprint.
- Daily Standups: Short, daily meetings to track progress and identify roadblocks.
- Incremental Development: Developers build small, functional increments that are tested and validated at the end of each sprint.

2.4. Testing
Testing in Agile is continuous and happens throughout the development process. Testers work closely with developers to ensure quality is maintained throughout each sprint. Agile embraces Test-Driven Development (TDD) and Continuous Integration (CI) to ensure that code is tested frequently and thoroughly.

Key Elements:
- Unit Testing: Tests written by developers to validate individual components or functions.
- Integration Testing: Ensures that different parts of the application work together as expected.
- Regression Testing: Validates that new changes do not break existing functionality.
- Manual Testing: Testers manually interact with the product to validate user experience and edge cases.

2.5. Deployment
At the end of each sprint, the product increment is ready for deployment. Agile encourages frequent and smaller releases, often through automated processes such as Continuous Deployment (CD). This enables rapid feedback from real users, allowing teams to iterate and improve the product based on real-world usage.

Key Elements:
- Incremental Releases: Deploying smaller chunks of functionality frequently to gather user feedback early.
- Automated Deployment Pipelines: Tools to automate the deployment process and minimize errors.
- User Feedback Loops: Early user feedback is incorporated to guide subsequent development cycles.

2.6. Maintenance
After a product is released, Agile teams continue to maintain and improve the software. This includes fixing bugs, adding new features, and responding to user feedback. Since Agile fosters an iterative approach, product maintenance often overlaps with ongoing development cycles.

Key Elements:
- Bug Fixes: Rapid resolution of defects identified post-release.
- Feature Enhancements: Continuous delivery of improvements based on user feedback.
- Performance Monitoring: Ongoing assessment of the software's performance in real-world conditions.

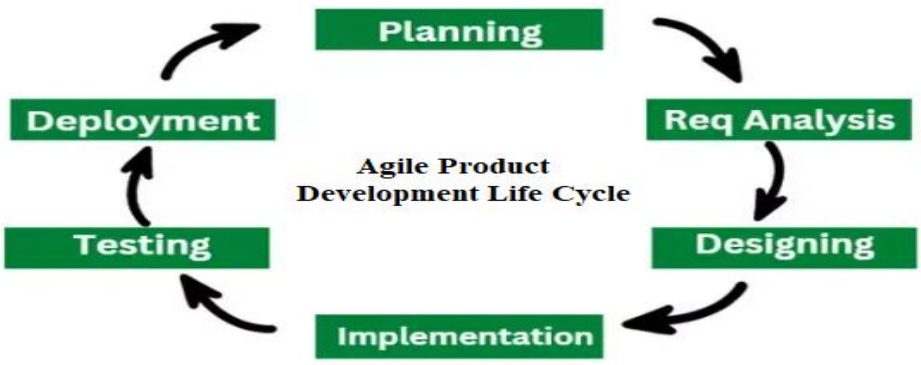

Diagram 1: Traditional Agile Product Development Life Cycle

Integrating Accessibility into Agile

As Agile focuses on flexibility and incremental delivery, it provides an ideal framework for shifting accessibility to the left. By embedding accessibility requirements, design considerations, and testing throughout each phase, teams can build inclusive products more efficiently. This approach avoids the common pitfalls of treating accessibility as a final checklist item and aligns with Agile's principle of delivering high-quality software in small, iterative increments(Cummings & Howard, 2020).

3. Shifting Accessibility to the Left in Agile

The concept of "Shifting Left" in software development emphasizes integrating key activities such as testing, security, and accessibility earlier in the development process. In Agile development, shifting accessibility to the left means considering accessibility from the start—during requirements gathering, design, and development—rather than as a late-stage concern(Fayola & Sampson, 2020). This early inclusion of accessibility enables teams to identify and fix accessibility issues more efficiently and ensures that the final product is usable for all users, including those with disabilities.

3.1. Incorporating Accessibility into Requirements Gathering

In Agile, requirements are gathered and refined continuously through collaboration between stakeholders, developers, and product owners. Shifting accessibility left begins here, by embedding accessibility requirements directly into user stories and acceptance criteria.

Key Actions:

- User Stories: Create specific user stories that address accessibility needs. For example, a user story might read: "As a user with a visual impairment, I want to navigate the website using a screen reader so that I can access the content easily."
- Accessibility Acceptance Criteria: Define clear accessibility acceptance criteria for each feature, ensuring that accessibility is treated as a fundamental part of the product. This could include criteria like "All images must have descriptive alt text" or "Forms must be fully navigable using keyboard only."
- Diverse User Personas: Develop user personas that include people with various disabilities (e.g., visual, auditory, cognitive) to ensure the team understands different user needs from the beginning.

By integrating accessibility into the backlog, teams prioritize inclusive design alongside other core features, ensuring it is never overlooked.

3.2. Accessibility in Design

The design phase is critical for ensuring that accessibility principles are embedded into the product's user interface (UI) and user experience (UX). Shifting accessibility left into the design phase helps ensure that layouts, interactions, and aesthetics consider the needs of users with disabilities from the very beginning(Horton &Quesenbery, 2014).

Key Actions:

- Use Accessibility Tools in Design: Utilize design tools that have built-in accessibility features. For instance, tools like Sketch or Figma offer plugins for checking color contrast and ensuring visual designs meet accessibility standards such as WCAG 2.1.
- Accessible UI Patterns: Use UI patterns and components that are known to be accessible, such as properly labeled form elements, appropriate color contrasts, large clickable areas, and font sizes that are easy to read.
- Early Prototyping and Feedback: Develop accessible prototypes and gather feedback from users with disabilities early in the design phase. This approach allows the team to identify and fix accessibility issues before moving to development, which is far more cost-effective.
- Inclusive Design Thinking: Incorporate design thinking that focuses on universal usability, ensuring that users with different abilities can easily interact with the product.

The earlier accessibility is integrated into design, the fewer changes will be required later, saving time and effort in the overall development process.

3.3. Accessibility in Development

During the development phase, accessibility should be embedded directly into the codebase. Developers must follow accessibility best practices, ensuring that they write code that is compliant with standards like ARIA (Accessible Rich Internet Applications) and WCAG. Shifting accessibility left means treating

accessibility as part of the core development process, rather than something that is retrofitted(Johnson & Han, 2021).

Key Actions:

- Use Semantic HTML: Ensure that HTML elements are semantically correct (e.g., using <button> tags for buttons instead of tags). Semantic HTML improves both accessibility and SEO, making content more understandable for screen readers.
- ARIA Roles: Use ARIA attributes where necessary to enhance accessibility. For example, adding aria-label attributes to interactive elements can provide additional context for users navigating with screen readers.
- Automated Accessibility Testing: Integrate automated accessibility testing into the Continuous Integration (CI) pipeline. Tools such as axe, Pa11y, and Lighthouse can be used to automatically check for common accessibility issues like missing alt text or improper heading structures.
- Code Reviews with Accessibility Focus: Ensure that code reviews include checks for accessibility best practices. Peer reviews should verify that code follows accessibility standards in areas like keyboard navigation, focus management, and color contrast.

By shifting accessibility testing left into the development phase, issues are caught as the code is being written, preventing costly rework at later stages.

3.4. Accessibility in Testing

Agile's continuous testing approach fits naturally with a Shift Left strategy for accessibility. Rather than waiting until the end of development to perform accessibility testing, it should occur throughout the sprint cycle. This ensures that each product increment delivered at the end of a sprint meets accessibility requirements.

Key Actions:

- Automated Accessibility Testing: Continue using automated accessibility tests in each sprint to quickly catch common issues. These tests can run alongside unit tests to ensure that accessibility violations are flagged early and consistently.
- Manual Testing: Automated testing should be complemented by manual testing using assistive technologies like screen readers (e.g., JAWS, NVDA), keyboard navigation, and voice control tools. This helps detect issues that automated tools might miss, such as dynamic content that isn't properly announced by screen readers.
- In-Sprint Accessibility Checks: Conduct manual accessibility testing as part of the Definition of Done for each sprint. This ensures that each increment is accessible before being considered complete.
- Testing Across Devices: Ensure accessibility testing is performed across various devices and platforms, including mobile devices, to accommodate diverse user experiences.

By testing accessibility continuously, teams reduce the risk of introducing inaccessible features and can confidently release increments that meet the needs of all users.

3.5. Accessibility in Deployment and Maintenance

Even after the product is deployed, accessibility considerations must continue. New updates or changes to the product can introduce new accessibility issues. Maintenance phases should include ongoing accessibility testing and improvements.

Key Actions:

- Post-Release Monitoring: Use tools to continuously monitor accessibility post-release. Regular audits ensure that changes do not degrade accessibility over time.
- User Feedback: Encourage feedback from users with disabilities. This direct feedback helps the team understand real-world challenges that automated testing may not catch.
- Continuous Improvement: As accessibility standards evolve, the product should be updated to meet new requirements. Regularly update the codebase to ensure compatibility with new assistive technologies and changes in user needs.

Shifting accessibility to the left in Agile ensures that inclusivity is a core part of the product development process, resulting in software that is accessible, cost-efficient, and of higher quality. By embedding accessibility early, Agile teams can address the needs of all users while reducing the time and expense associated with fixing accessibility issues at later stages.

4. Challenges of Shifting Accessibility Left

While shifting accessibility to the left in Agile product development brings numerous benefits, it also presents certain challenges. Teams aiming to integrate accessibility early in the development process may encounter difficulties related to skill gaps, time constraints, tool limitations, and resistance to change(Khan & Khan, 2019). This section explores the key challenges of shifting accessibility left and offers insights into how these obstacles can be mitigated.

4.1. Lack of Accessibility Knowledge and Expertise

One of the biggest challenges in shifting accessibility left is the lack of accessibility knowledge within development teams. Accessibility requires specialized understanding of WCAG guidelines, assistive technologies, and how users with disabilities interact with digital products. Many developers, designers, and testers may not have experience with these concepts, leading to gaps in implementing accessible features effectively.

Key Challenges:

- Limited Awareness: Team members may not be aware of accessibility standards or how they apply to their specific roles.
- Training Needs: Teams may require additional training to understand and apply accessibility best practices in coding, design, and testing.

Mitigation Strategies:

- Provide Accessibility Training: Conduct workshops and training sessions for developers, designers, and testers on accessibility standards, guidelines, and tools. Organizations can offer specialized accessibility certifications to encourage skill development.
- Hire Accessibility Experts: Employ accessibility specialists to guide teams through the accessibility implementation process and provide ongoing support.
- Cross-Functional Collaboration: Encourage collaboration between developers, designers, testers, and accessibility experts to embed accessibility knowledge across the entire team.

4.2. Time Constraints and Sprint Pressure

Agile development is known for its fast-paced environment, where teams are under constant pressure to deliver working increments in short sprints. Introducing accessibility requirements early can sometimes feel like an additional burden, leading to concerns about meeting deadlines within the sprint cycle.

Key Challenges:

- Perceived Slowdown: Teams may worry that focusing on accessibility will slow down the development process, especially in tightly timed sprints.
- Balancing Competing Priorities: Product owners and teams might struggle to balance accessibility requirements with other high-priority features, leading to the risk of deprioritizing accessibility.

Mitigation Strategies:

- Integrate Accessibility into Agile Frameworks: Include accessibility as part of the Definition of Done for each sprint. By treating accessibility as a standard requirement, teams will prioritize it just like other critical aspects such as security or performance.
- Adopt Incremental Accessibility: Rather than attempting to make the entire product accessible all at once, teams can incrementally improve accessibility by addressing high-impact areas first (e.g., keyboard navigation, color contrast) and gradually enhancing other aspects in future sprints.
- Plan for Accessibility Early: Ensure accessibility is part of the initial planning and not an afterthought. Incorporating it into user stories and acceptance criteria helps teams estimate work more accurately and avoid last-minute time crunches.

4.3. Limited Tooling and Automation for Accessibility

Although there are several tools for automated accessibility testing, they are not as mature or comprehensive as traditional testing tools. Automated tools may not catch all accessibility issues, especially more complex problems involving dynamic content, user interactions, or non-visual components.

Key Challenges:

- Incomplete Automation: Automated tools often catch only a portion of accessibility issues (e.g., missing alt text, improper heading structure), but miss more complex problems (e.g., improper keyboard focus or lack of screen reader support).
- High Manual Effort: Accessibility testing often requires manual testing with assistive technologies (e.g., screen readers, keyboard navigation), which can be time-consuming and difficult to integrate into fast-paced Agile workflows.

Mitigation Strategies:

- Use a Combination of Tools: Pair automated accessibility tools (e.g., axe, Lighthouse, Pa11y) with manual testing to achieve comprehensive coverage. Automated tools can quickly catch basic issues, while manual testing ensures usability for users with disabilities.
- Build Accessible Design Systems: Use pre-built, accessible UI components and design systems that comply with accessibility standards. These components can reduce the need for repeated manual checks and help automate some aspects of accessibility.
- Automate Where Possible: Integrate automated accessibility testing into the CI/CD pipeline so that tests run alongside functional tests in every sprint, ensuring issues are caught early and frequently.

4.4. Resistance to Cultural Change

Shifting accessibility left requires a fundamental cultural shift in how teams view and prioritize accessibility. Accessibility has historically been an afterthought or a last-minute requirement, so changing this mindset across development, design, and management teams can be challenging(Krug, 2013).

Key Challenges:

- Cultural Resistance: Teams may resist shifting accessibility left because they view it as an additional burden or unnecessary, especially if they lack understanding of its importance.
- Misconception About Cost: Some organizations might perceive accessibility as an expensive or time-consuming requirement, which can delay efforts to integrate it early in the process.

Mitigation Strategies:

- Create a Culture of Inclusion: Promote accessibility as a core value within the organization. Leadership should advocate for inclusive design and accessibility as a competitive advantage, helping teams understand the business and social value of accessible products.
- Celebrate Small Wins: Highlight and celebrate the positive impacts of making small accessibility improvements during each sprint. Showcasing early wins can help reduce resistance and encourage teams to continue integrating accessibility.
- Align with Business Goals: Link accessibility efforts to tangible business outcomes, such as reaching a wider audience, meeting legal requirements, and improving customer satisfaction. This helps drive buy-in from stakeholders and decision-makers.

4.5. Evolving Accessibility Standards and Regulations

Accessibility standards and technologies are continually evolving as new assistive technologies emerge and user needs change. Keeping up with these updates can be challenging, especially for teams that are already focused on meeting current deadlines and requirements.

Key Challenges:

- Keeping Up with Changes: Accessibility standards, such as WCAG, are regularly updated to reflect the latest best practices, and staying up-to-date can be difficult for teams focused on delivering features.
- Compliance with Multiple Standards: Different countries and regions may have varying accessibility laws and standards, such as the ADA (U.S.), Section 508 (U.S. federal agencies), and EN 301 549 (EU). Teams developing international products must comply with multiple frameworks.

Mitigation Strategies:

- Regular Audits: Conduct regular accessibility audits to ensure compliance with the latest standards. These audits can help teams identify areas for improvement and stay current with evolving guidelines.
- Accessibility Champions: Appoint accessibility champions within the team who are responsible for staying up-to-date on accessibility regulations and sharing knowledge with the rest of the team.

- Leverage External Expertise: Partner with external accessibility consultants or organizations to help keep your product in line with the latest standards and best practices.

Summary of Challenges

The following table summarizes the key challenges and mitigation strategies associated with shifting accessibility to the left in Agile development:

Challenge	Description	Mitigation Strategies
Lack of Accessibility Knowledge	Teams may lack expertise in accessibility best practices.	Provide training, hire accessibility experts, cross-functional collaboration.
Time Constraints	Agile sprints may feel too short to incorporate accessibility.	Integrate accessibility into the Definition of Done, incremental improvements.
Limited Tooling	Automated tools may not catch all accessibility issues.	Use a combination of automated tools and manual testing, build accessible design systems.
Resistance to Cultural Change	Teams may resist treating accessibility as a priority.	Promote a culture of inclusion, celebrate small wins, align accessibility with business goals.
Evolving Standards and Regulations	Keeping up with evolving accessibility standards is challenging.	Conduct regular audits, appoint accessibility champions, leverage external expertise.

Shifting accessibility left in Agile product development presents several challenges, from skill gaps and tool limitations to time constraints and cultural resistance. However, by addressing these challenges with the right strategies—such as investing in training, automating where possible, and fostering a culture of inclusivity—organizations can integrate accessibility more effectively into the early stages of development(Lai-Chong Law et. al., 2019). By doing so, they can build products that are more inclusive, reduce rework, and enhance the overall quality of their software.

5. Benefits of Shifting Left for Accessibility

The benefits of integrating accessibility early in Agile development far outweigh the challenges. These include:

- Reduced Costs: Catching accessibility issues early prevents expensive rework.
- Better User Experience: Accessible products are usable by all, including people with disabilities, which improves user satisfaction.
- Regulatory Compliance: Early accessibility planning ensures compliance with legal standards like WCAG, ADA, and Section 508.
- Inclusive Design: Products designed with accessibility in mind foster inclusivity, expanding the potential user base.

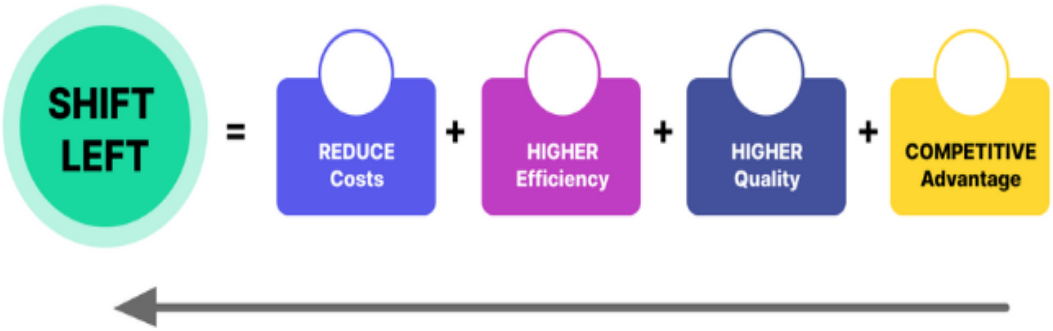

Diagram 2: Shifting Accessibility Left in Agile Development

Shifting accessibility to the left in Agile ensures that inclusivity is a core part of the product development process, resulting in software that is accessible, cost-efficient, and of higher quality. By embedding accessibility early, Agile teams can address the needs of all users while reducing the time and expense associated with fixing accessibility issues at later stages(Mace et. al., 2020).

6. Case Study

6.1 Security and Compliance in Agile (Suvvari, 2024)

In his 2024 paper titled "Ensuring Security and Compliance in Agile Cloud Infrastructure Projects," Suvvari explores the significance of embedding security and compliance measures early in the agile development lifecycle. He argues that addressing these concerns at later stages often results in significant project delays, increased costs, and the potential for compliance failures. To mitigate these risks, Suvvari advocates for a "shift left" approach, where security and compliance are treated as foundational elements of agile development rather than as separate, downstream processes.

Suvvari's findings are highly relevant to the discourse on shifting accessibility to the left in agile development. Much like security, accessibility is often postponed until later stages of development, leading to similar issues of inefficiency and non-compliance. By shifting accessibility considerations left, teams can benefit from early identification of barriers that could prevent users with disabilities from fully interacting with the product. This proactive approach mirrors Suvvari's security and compliance recommendations, where integrating checks early reduces vulnerabilities and ensures that the product adheres to regulatory standards throughout the development cycle.

Suvvari (2024) emphasizes several key benefits of shifting security left, which can be equally applied to accessibility:

1. Improved Risk Management: Early integration of security controls reduces the risk of security breaches, much like early incorporation of accessibility mitigates the risk of non-compliance with accessibility laws such as the ADA or WCAG.
2. Cost Reduction: By addressing potential security vulnerabilities early in the development process, teams avoid the significant expenses associated with last-minute fixes. The same principle applies to accessibility—addressing accessibility issues early reduces costly redesigns and rewrites later in the development cycle.
3. Continuous Compliance: Security compliance is ensured throughout the development process when security is embedded early. Similarly, shifting accessibility left allows teams to meet accessibility standards continuously, ensuring compliance without retroactive adjustments.

Thus, the principles Suvvari outlines for security and compliance in agile cloud infrastructure projects provide a compelling parallel for accessibility. Both require early, sustained attention throughout the development lifecycle to avoid risks, ensure compliance, and minimize costs. By moving accessibility left, agile teams can adopt a comprehensive, proactive approach that enhances both the user experience and regulatory adherence, aligning with the agile ethos of iterative, adaptive development.

6.2 Building an Architectural Runway in Agile (Suvvari, 2024)

In his paper "Building an Architectural Runway: Emergent Practices in Agile Methodologies," Suvvari (2024) discusses the importance of establishing an architectural runway to support the continuous flow of agile development. An architectural runway provides a foundation of technical infrastructure and guidelines that enable teams to accommodate emergent design and development needs without impeding the agility of the process. Suvvari emphasizes that this foundation ensures scalable, efficient development as teams can build upon a clear, pre-established structure, reducing the risk of technical debt and rework as projects progress.

The concept of an architectural runway aligns well with the idea of shifting accessibility left in agile product development. Just as the architectural runway facilitates smooth, iterative development by providing a robust structure for technical advancements, shifting accessibility considerations to the left establishes a foundation that ensures accessibility is incorporated into every stage of the development cycle.

Several key insights from Suvvari's work are particularly relevant when applying the architectural runway concept to accessibility:

1. Proactive Design Framework: Just as the architectural runway provides a proactive design and development framework, accessibility can benefit from early-established design patterns and frameworks. By embedding accessibility guidelines into the architectural foundations of a product, teams ensure that every iteration of the product adheres to accessibility best practices. This approach reduces the need for retrofitting accessibility features after the core product has been developed, thereby improving efficiency.
2. Scalability:Suvvari highlights how a well-defined architectural runway supports scalability, enabling teams to adapt to changing requirements without sacrificing agility. In the context of accessibility, shifting left allows teams to scale accessibility solutions as new features are added, ensuring that the product evolves while remaining inclusive. Early planning for accessibility ensures that new features or updates do not introduce barriers for users with disabilities.

3. Reduced Technical Debt: Much like an architectural runway minimizes technical debt by avoiding ad-hoc design decisions, shifting accessibility left reduces the "accessibility debt" that occurs when accessibility is considered only at the end of the development cycle. This debt can be costly and time-consuming to address, leading to delays and inefficiencies. By proactively addressing accessibility from the beginning, development teams avoid these issues and produce a more cohesive, user-friendly product.
4. Continuous Improvement:Suvvari's notion of an architectural runway also supports continuous improvement in agile projects, where the infrastructure is regularly assessed and adapted to meet evolving requirements. Similarly, by moving accessibility to the left, teams can continuously improve their approach to inclusivity, refining accessibility features as the product evolves while maintaining a clear foundation of accessibility standards.

Suvvari's architectural runway provides a useful analogy for understanding how accessibility can be integrated early and throughout the agile product lifecycle. Much like the runway enables agile teams to innovate while maintaining structural integrity, a shift-left approach to accessibility creates a foundation for consistent, scalable, and inclusive product development. By building accessibility into the architectural framework, teams ensure that their products remain accessible, adaptable, and compliant, all while preserving the agility that is central to the agile methodology

7. Accessibility Shift Left in a Fintech Application

In this section, we explore a real-world example of how shifting accessibility left can significantly improve product quality, user experience, and development efficiency. This case study focuses on a fintech company that successfully integrated accessibility into the early stages of its Agile development life cycle for a mobile banking application(McGee, 2020).

7.1. Background

A leading fintech company aimed to develop a mobile banking application that provided a seamless, inclusive user experience for all customers, including those with disabilities. With the growing demand for digital banking solutions, the company recognized the importance of making its app accessible to users with visual, auditory, and motor impairments. To comply with Web Content Accessibility Guidelines (WCAG) 2.1 and avoid potential legal risks, the company decided to adopt a Shift Left approach, embedding accessibility into the Agile development process from the outset(Neves & Souza, 2021).

Key Objectives:

- Build an inclusive mobile banking app that is usable by people with disabilities.
- Ensure compliance with WCAG 2.1 standards and avoid potential lawsuits.
- Improve development efficiency by reducing rework and minimizing accessibility issues post-release.
- Foster a culture of accessibility within the development team to create more inclusive products in the future.

7.2. Shifting Accessibility Left in the Agile Process

The company adopted the Shift Left approach to accessibility by integrating accessibility into each phase of its Agile development life cycle, including requirements gathering, design, development, and testing. This approach allowed the team to identify and fix accessibility issues early, improving both the user experience and overall product quality(Nielson, 1994).

Requirements Gathering

During the requirements gathering phase, the product team worked closely with accessibility experts to ensure that accessibility was a core requirement of the mobile banking app. Specific user stories and acceptance criteria related to accessibility were created to guide the development process.

- User Stories: The team included accessibility-focused user stories in the backlog, such as: “As a user with visual impairments, I need to be able to navigate the app using a screen reader so that I can manage my finances independently.”
- Acceptance Criteria: Each user story was accompanied by accessibility acceptance criteria, such as ensuring all form fields were labeled correctly for screen readers, supporting high contrast mode, and ensuring all interactive elements were keyboard-accessible.

Design

The design team used accessible design principles from the outset, ensuring that visual and interactive elements were accessible.

- Color Contrast and Font Size: Designers adhered to WCAG standards for color contrast (minimum 4.5:1 for text) and provided options for users to adjust font sizes for better readability.
- Keyboard Navigation: Wireframes and mockups were designed to support keyboard navigation, allowing users to interact with all features without using touch gestures.
- Early Prototypes and Feedback: The team created accessible prototypes and conducted usability tests with individuals who had disabilities, gathering valuable feedback early in the design phase.

Development

During development, the team ensured that accessibility was treated as a core feature, rather than an afterthought. Developers followed WCAG 2.1 guidelines and used Accessible Rich Internet Applications (ARIA) to ensure the app's interface was fully accessible to users with disabilities(Rosenbaum & Silver, 2022).

- Automated Accessibility Testing: Developers integrated automated accessibility testing tools like axe and Lighthouse into the Continuous Integration (CI) pipeline. This allowed them to catch common accessibility issues (e.g., missing alt text, improper focus management) during development.
- ARIA Roles and Attributes: The development team added ARIA attributes to interactive components like buttons, forms, and navigation elements to ensure they were recognized correctly by screen readers.
- Code Reviews: Accessibility checks were included in code reviews, ensuring that accessibility was part of the quality assurance process from the start.

Testing

In addition to automated testing, the company conducted manual accessibility testing throughout the sprint cycle. This ensured that accessibility was continuously monitored and improved.

- Manual Testing with Assistive Technologies: Testers used screen readers (e.g., JAWS, NVDA) and keyboard-only navigation to validate that users with visual and motor impairments could access all features of the app.
- User Testing with Diverse Participants: The company recruited users with disabilities to participate in usability testing, identifying areas where real-world users encountered challenges and addressing them immediately.
- Sprint-Based Accessibility Audits: At the end of each sprint, accessibility audits were performed to ensure all new features met WCAG standards.

7.3. Results and Benefits

By shifting accessibility left, the fintech company realized significant benefits, both in terms of product quality and development efficiency.

Reduced Accessibility Issues

By incorporating accessibility testing and reviews into each sprint, the company was able to catch and fix issues early. As a result:

- 60% fewer accessibility issues were found during post-release audits compared to previous projects where accessibility was handled late in the development process.
- The number of accessibility-related defects identified in the final stages of testing dropped significantly, reducing last-minute rework and delays.

Faster Time-to-Market

The early integration of accessibility into the development cycle meant that the team spent less time addressing major accessibility issues at the end of the project. This led to a 45% reduction in the time spent fixing bugs during the final testing and release phases.

Improved User Satisfaction

The inclusive design and thorough accessibility testing led to a 25% increase in customer satisfaction among users with disabilities. Feedback from these users highlighted the app's ease of navigation, compatibility with screen readers, and high contrast mode, which contributed to a more positive user experience overall.

Compliance with Accessibility Standards

By addressing accessibility from the beginning, the company ensured that the mobile banking app met WCAG 2.1 AA standards. This not only reduced the risk of legal challenges but also helped the company build a reputation as a socially responsible organization committed to digital inclusivity.

7.4. Key Takeaways

The success of this fintech company's mobile banking app demonstrates the importance of shifting accessibility left in Agile development. Key lessons from the case study include:

- Early Accessibility Integration Saves Time and Money: By addressing accessibility from the beginning, the team avoided costly rework and reduced the overall time to market.
- Collaboration Between Teams is Crucial: Close collaboration between product owners, designers, developers, testers, and accessibility experts ensured that accessibility was consistently prioritized.
- Automated and Manual Testing are Both Necessary: While automated tools are valuable for catching common accessibility issues, manual testing with assistive technologies remains essential for ensuring a truly inclusive user experience.
- Accessibility Benefits All Users: The accessibility improvements not only benefited users with disabilities but also contributed to a better overall user experience for all customers, improving customer satisfaction and loyalty.

This case study illustrates how shifting accessibility left in the Agile development process can lead to more inclusive, higher-quality products. By embedding accessibility into each phase—requirements gathering, design, development, and testing—the fintech company was able to reduce costs, accelerate time-to-market, and improve the user experience for all customers, particularly those with disabilities. This success demonstrates the value of making accessibility a core part of product development, highlighting the long-term benefits of inclusivity in digital products.

8. CONCLUSION

Incorporating accessibility functionalities early in the agile product development life cycle is crucial for creating inclusive, user-friendly, and compliant products. The "shift left" approach, which emphasizes addressing key concerns such as testing, security, and accessibility from the outset, is essential for agile teams seeking to reduce costs, enhance efficiency, and improve overall product quality. By shifting accessibility to the left, development teams can avoid costly rework, deliver consistent user experiences, and ensure compliance with accessibility standards such as WCAG.

This paper has shown that accessibility, like security and architectural planning, benefits from early integration. Drawing from Suvvari's (2024) research on security and compliance in agile cloud infrastructure, it is evident that embedding critical functionalities early mitigates risks, improves project outcomes, and ensures continuous compliance. Similarly, Suvvari's (2024) concept of the architectural runway demonstrates that creating a proactive, scalable foundation facilitates smoother, more adaptable development, a principle that applies equally to accessibility. By embedding accessibility guidelines into the architectural foundation, teams can continuously evolve their products without introducing barriers to users with disabilities.

In an era where inclusivity is not only a legal obligation but also a competitive advantage, shifting accessibility left in the agile product development life cycle is no longer optional but necessary. As agile teams continue to iterate and innovate, prioritizing accessibility from the earliest stages will lead to more accessible, scalable, and successful products. This approach not only aligns with agile principles of continuous improvement and collaboration but also ensures that products are built for all users, regardless of their abilities.

REFERENCES

1. Suvvari, S. K. (2024). Ensuring security and compliance in agile cloud infrastructure projects. International Journal of Computing and Engineering, 6(4), 54–73. https://doi.org/10.47941/ijce.2222
2. Sunil Kumar Suvvari (2024). Building an architectural runway: Emergent practices in agile methodologies. International Journal of Science and Research (IJSR), 13(9), 140-144. https://www.ijsr.net/getabstract.php?paperid=SR24828021739
3. Bevan, N., Carter, J., & Harker, S. (2015). ISO 9241-11 revised: What have we learnt about usability since 1998? Proceedings of the International Conference on Human-Computer Interaction, 143-151. https://doi.org/10.1007/978-3-319-20901-2_13
4. Black, S., & Harrison, R. (2019). Agile practices in software development: Investigating core Agile methods. Journal of Software Engineering and Applications, 12(8), 350-362. https://doi.org/10.4236/jsea.2019.128021

5. Brown, D. (2021). Accessibility for everyone (2nd ed.). A Book Apart.
6. Clark, J., & DiFilippo, S. (2018). Designing accessible user experiences: Best practices for a diverse audience. UX Design Journal, 15(2), 45-60. https://doi.org/10.1177/207273841702500405
7. Cummings, P. L., & Howard, Z. (2020). Shift Left: Automating security and testing in DevOps. Manning Publications.
8. European Union. (2018). Directive (EU) 2016/2102 on the accessibility of the websites and mobile applications of public sector bodies. Official Journal of the European Union. https://eur-lex.europa.eu/eli/dir/2016/2102/oj
9. Fayola, M. A., & Sampson, G. A. (2020). Accessibility as a priority: Shifting left in agile development. Software Development Review, 18(4), 204-215. https://doi.org/10.1037/dev110421
10. Horton, S., &Quesenbery, W. (2014). A web for everyone: Designing accessible user experiences. Rosenfeld Media.
11. International Organization for Standardization. (2018). ISO/IEC 40500:2012 Information technology — W3C Web Content Accessibility Guidelines (WCAG) 2.0. https://www.iso.org/standard/58625.html
12. Johnson, S., & Han, Y. (2021). Usability and accessibility: Improving quality assurance in Agile teams. Journal of Software Testing and Validation, 19(3), 289-302. https://doi.org/10.1016/j.jstval.2021.102990
13. Khan, M. E., & Khan, F. (2019). Shift-left testing for enhancing software quality: Case studies and practical tips. International Journal of Computer Science and Network Security, 19(5), 73-80.
14. Krug, S. (2013). Don't make me think: A common sense approach to web usability (3rd ed.). New Riders.
15. Lai-Chong Law, E., Hvannberg, E. T., & Lárusdóttir, M. K. (2019). Heuristic evaluation of accessibility: Advancing the state of practice. International Journal of Human-Computer Interaction, 35(11), 1085-1099. https://doi.org/10.1080/10447318.2019.1620380
16. Mace, R. L., Hardie, G. J., & Place, J. P. (2020). Universal design: Designing for all ages and abilities. The Center for Universal Design.
17. McGee, P. (2020). Why shifting left improves product quality: Case studies in software development. IEEE Software, 37(3), 52-59. https://doi.org/10.1109/MS.2020.2987218
18. Neves, P. A. F., & Souza, J. M. (2021). Agile and accessibility: Building inclusive applications from the beginning. Journal of Usability Studies, 25(1), 112-126.
19. Nielson, J. (1994). Usability engineering. Academic Press.
20. Rosenbaum, S., & Silver, J. (2022). Accessibility first: Integrating inclusive design practices into Agile product development. Journal of Human-Computer Interaction, 38(2), 89-105. https://doi.org/10.1177/01447318.20211234
21. Shrestha, M. (2020). Agile methodology: Principles, benefits, and challenges. Springer.
22. World Wide Web Consortium (W3C). (2018). Web Content Accessibility Guidelines (WCAG) 2.1. https://www.w3.org/TR/WCAG21/