AALBORG UNIVERSITY
Fall 2025

Agile Software Engineering

EXAM ASSIGNMENT

STUDENT

Jawad Mehmood Khan Qayyum

20233947

jqayyu23@student.aau.dk

INTERVIEWEE

Ahmed Sadiq

Djøf Trade Union

Asq@djoef.dk

PROJECT PERIOD
2025

AALBORG UNIVERSITY
STUDENT REPORT

AALBORG UNIVERSITY

Title
Agile Software Engineering

Project period
2025

Student
Jawad Mehmood Khan Qayyum

Semester
Fall 2025

Number of pages
42

Synopsis

Based on an interview with a senior developer at Djøf Trade Union, this report examines how a small internal team adapts Agile methods to fit their organizational context. The analysis covers their pragmatic approach to Scrum, refinement workflows, quality assurance processes, and how they balance stakeholder expectations with technical constraints.
The report uses course concepts to interpret the interview data, identifying practical trade-offs the team makes when applying Agile principles and offering insights into what works for teams transitioning from legacy systems.

Contents

1 Introduction

This report is part of the exam assignment in the Agile Software Engineering course at Aalborg University Copenhagen. The aim is to analyze how Agile principles are implemented in practice and assess their impact on project success. For this purpose, I interviewed Ahmed Sadiq, a software developer with over 10 years of experience at Djøf Trade Union. Ahmed works in the internal development department, where he builds systems like booking platforms, account management tools, and APIs serving both employees and union members.

The organization faced challenges with legacy systems and unclear processes before gradually adopting Scrum. Their approach is not strictly by-the-book: the team stopped using story points after stakeholders found them confusing, and they sometimes skip retrospectives when a sprint has little to discuss. This adaptive approach aligns with industry trends where teams modify Agile frameworks to fit their context rather than following rigid prescriptions *(State of Agile Report)*.

This report examines their Agile practices, covering refinement processes, quality assurance, deployment pipelines, and stakeholder management, using concepts from the course to connect theory with practice.

2

Recognizing Development Challenges

Before Djøf Trade Union adopted Scrum, their development team faced significant challenges that hindered their ability to deliver value efficiently. Understanding these problems is essential to appreciate why they chose Agile practices and how those practices addressed their specific needs. The Scrum framework emphasizes that transparency enables inspection and adaptation *(Schwaber and Sutherland 2020)*, but achieving transparency first requires identifying what is unclear. Research with 2,000 Scrum teams found that team effectiveness depends heavily on addressing fundamental organizational impediments *(Verwijs and Russo 2021)*.

2.1 Key Challenges

Djøf Trade Union faced a fundamental problem with how work arrived and was prioritized. Tasks would emerge ad-hoc without clear consideration for team capacity or strategic importance, creating an environment where developers constantly reacted to incoming requests rather than working proactively toward meaningful goals.

The situation became particularly problematic when dealing with legacy systems. Ahmed described a project where he was asked to add a simple checkbox to the member registration system for union unemployment insurance sign-up **(Transcript: 00:02:23–00:02:34)**. The legacy membership system had security restrictions that prevented making changes without being logged in as an actual member, and creating test members was not permitted for data protection reasons **(Transcript: 00:03:04–00:03:33)**. He spent two weeks to a month working on this task, only to have it completely abandoned **(Transcript: 00:04:32–00:04:48)**.

The task should never have reached Ahmed in the first place. It needed approval from the legal department regarding GDPR compliance and coordination with the third-party vendor who controlled the membership system's security settings **(Transcript: 00:04:48–00:05:17)**. There was no gate-keeping process to ensure that tasks were actually achievable within existing constraints before developers invested time in them. This pattern repeated itself across various projects. Developers would receive assignments that looked simple on the surface but contained hidden dependencies, unclear stakeholder expectations, or technical constraints that had not been discovered.

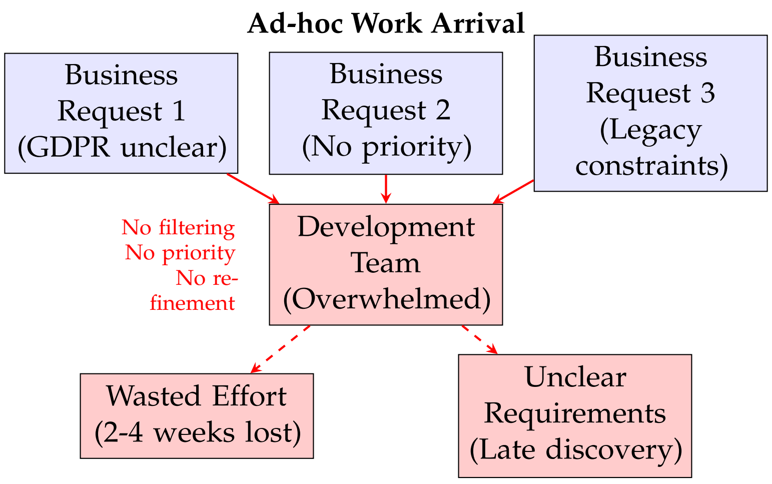
Ad-hoc Work Arrival
Business Request 1 (GDPR unclear)
Business Request 2 (No priority)
Business Request 3 (Legacy constraints)
No filtering
No priority
No refinement
Development Team (Overwhelmed)
Wasted Effort (2-4 weeks lost)
Unclear Requirements (Late discovery)

Figure 2.1: *Chaotic work arrival before Scrum adoption. Business requests flowed directly to developers without proper refinement, prioritization, or feasibility analysis, leading to wasted effort and late discovery of blockers.*

2.2 Problem Analysis

The core issue was the absence of structured processes to connect business needs with development capacity. When the business side wanted something built, there was no intermediary to translate those desires into well-defined, technically feasible requirements. Business stakeholders assumed their requests were feasible. Developers accepted tasks without fully understanding the business context or questioning whether all necessary approvals were in place. Both sides operated with incomplete information.

Without a clear backlog or product owner role, everything seemed equally urgent. There was no mechanism to evaluate trade-offs or make conscious decisions about what should be done first based on business value and technical dependencies. Ahmed's team also struggled with legacy code maintenance: systems built years earlier by developers no longer with the company, with sparse documentation **(Transcript: 00:16:07–00:16:37)**. This hidden complexity was rarely accounted for when estimating work.

2.3 Root Causes and Readiness for Change

The fundamental problem was the absence of empirical process control. Djøf Trade Union made decisions based on assumptions rather than validated information. None of these assumptions were regularly tested against reality until it was too late. Teams didn't have regular checkpoints to demonstrate what had been built and gather feedback. There were no retrospectives where teams could reflect on what went wrong and agree on changes to prevent similar issues in the future.

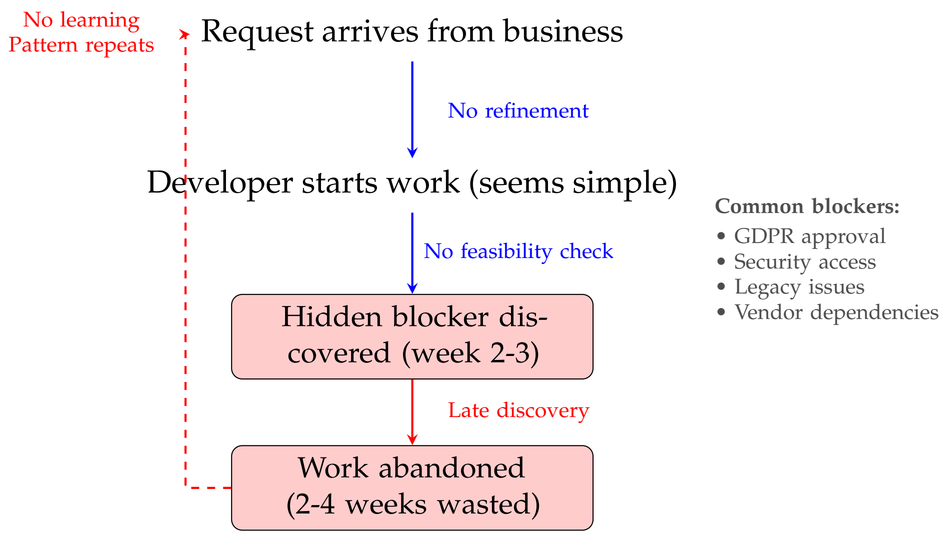

Figure 2.2: *Wasted effort cycle before Scrum adoption. Work requests arrived without refinement or feasibility checks, developers started tasks that seemed simple, hidden blockers emerged late in development, and work was abandoned after 2-4 weeks of effort. Without retrospectives or process changes, this pattern repeated continuously.*

Part of the root cause was also the absence of dedicated roles with clear accountability. No one owned the product backlog or facilitated the development process. They also lacked transparency into their capacity and work in progress. Without a visible backlog or sprint board, stakeholders couldn't see what the team was working on, creating frustration on both sides.

Finally, there was insufficient management support for changing how work was done. Research on Agile transformations demonstrates that management support is the second most critical success factor, creating conditions that enable developer effectiveness and continuous improvement to flourish *(Russo 2021)*. Without executive buy-in and resources dedicated to process improvement, teams had little leverage to push for changes.

These root causes (lack of empirical process control, absence of defined roles, insufficient transparency, and limited management support) combined to create an environment where software development was chaotic and unpredictable. However, recognizing these problems created readiness for change. Failed projects like the membership checkbox task made the costs of their current approach visible to stakeholders. This awareness became the catalyst for exploring structured frameworks - precisely the context that led Djøf Trade Union to explore Scrum as a potential solution.

3

Adopting Agile Methodologies

Djøf Trade Union addressed their development challenges by adopting Scrum practices that introduced structure, transparency, and empirical process control. The Scrum framework emphasizes empiricism through transparency, inspection, and adaptation *(Schwaber and Sutherland 2020)*. Research demonstrates that successful Agile adoption requires teams to adapt frameworks to their organizational realities *(Verwijs and Russo 2021)*. This chapter examines how they implemented Scrum practices to establish predictability and deliver value more consistently.

3.1 Establishing Product Backlog Refinement

The introduction of structured refinement sessions fundamentally changed how work flowed into the development team. Before refinement became routine, tasks arrived ad-hoc without proper analysis of feasibility or business value. The refinement process gave the team a mechanism to break down large initiatives systematically and identify issues before developers invested time. Product Backlog refinement is an ongoing activity where the team adds detail, estimates, and order to backlog items *(Schwaber and Sutherland 2020)*.

Ahmed described how refinement works: before starting any project, the team runs refinement sessions where they break down epics into user stories **(Transcript: 00:05:30–00:06:00)**. The team takes business needs from stakeholders and decomposes them into manageable tasks. This allows them to deliver features incrementally. When the business side has done their preparation well and the team conducts thorough refinement where each user story has a clear purpose, projects tend to succeed **(Transcript: 00:06:00–00:06:24)**.

This practice directly addressed the legacy system problem Ahmed encountered earlier. The checkbox task that consumed weeks should never have reached the development team without first verifying legal approval and vendor accommodation. Refinement created a gate-keeping process that prevented such situations. They now question assumptions, identify dependencies, and validate feasibility during refinement rather than discovering blockers mid-implementation. The refinement process also improved transparency for stakeholders. When epics broke down into twenty user stories, stakeholders gained realistic insight into the complexity involved.

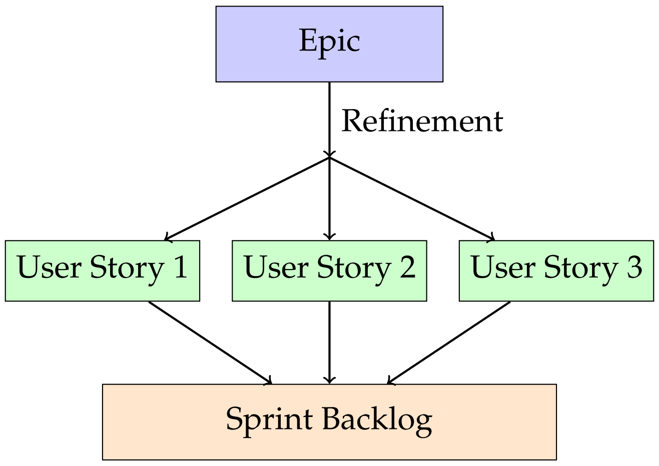

Figure 3.1: *Product Backlog Refinement Process: epics are decomposed into manageable user stories through collaborative refinement sessions.*

3.2 Introducing Scrum Roles and Sprint Structure

The adoption of defined Scrum roles brought accountability and clarity. Scrum defines three accountabilities: Product Owner, Scrum Master, and Developers, each with specific responsibilities for delivering value *(Schwaber and Sutherland 2020)*. The Product Owner became responsible for prioritizing the backlog in collaboration with the business, evaluating which work delivers the most value while considering technical dependencies **(Transcript: 00:14:52–00:15:22)**. This resolved the previous situation where everything seemed equally urgent. The Scrum Master role focused on maintaining the process and removing obstacles **(Transcript: 00:16:52–00:17:22)**.

Djøf Trade Union adopted two-week sprints after experimenting with different lengths. Three-week sprints proved too long, while one-week sprints were too short to deliver meaningful increments **(Transcript: 00:14:07–00:14:37)**.

The two-week cadence provided sufficient time to complete user stories while maintaining rapid feedback cycles.

The team adapted Scrum pragmatically. They dropped story points because the concept proved difficult to communicate consistently *(Russo 2021)*. Scrum Masters experimented with alternative estimation approaches to move away from estimates that stakeholders mistakenly interpreted as time commitments **(Transcript: 00:08:58–00:09:27)**. The team also occasionally skipped retrospectives when a sprint lacked meaningful progress to reflect on **(Transcript: 00:09:59–00:10:22)**, demonstrating their practical approach.

Sprint reviews after each sprint gave the team regular checkpoints to evaluate whether they succeeded and learn from failures **(Transcript: 00:06:48–00:07:31)**. Retrospectives provided space to refine how the team worked together. These ceremonies established inspection and adaptation cycles that were completely absent previously.

3.3 Building Collaboration and Transparency

Scrum practices fundamentally improved transparency. The visible backlog allowed stakeholders to see what the team was working on and understand capacity constraints. When stakeholders wanted new features, they could observe how those requests fit into existing priorities rather than assuming development resources were unlimited.

The sprint structure created predictability. Stakeholders knew when to expect completed features. The flexibility Ahmed mentioned as the best aspect of their approach stems from having a structured process that can accommodate change **(Transcript: 00:15:37–00:15:52)**. When business priorities shift, the team can adjust in the next sprint planning session.

Code review processes and the Azure DevOps pipeline with automated testing provided technical safeguards **(Transcript: 00:12:52–00:13:22)**. Automated tests run before code can merge. A colleague must review and approve the code before it progresses. The deployment pipeline moves code through development, test, and production environments systematically, creating quality gates at each stage. Delivering a Done increment every Sprint reduces the risk of undone work accumulating and disrupting future development *(Verwijs, Overeem, et al. 2020)*.

Collaboration within the team also improved. Developers could focus on defined sprint goals rather than constantly reacting to incoming requests. The Product Owner acted as a buffer, managing stakeholder expectations and protecting the team from excessive interruptions. This allowed developers to work more proactively and build solutions with proper consideration for technical quality.

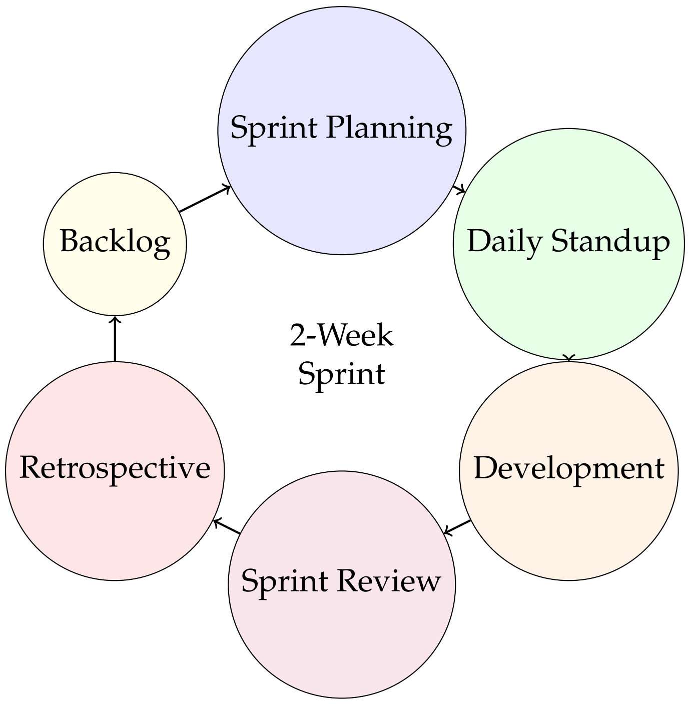

Figure 3.2: *Sprint Cycle: Ahmed's team follows a two-week sprint cadence with regular ceremonies for planning, coordination, review, and continuous improvement.*

3.4 Addressing Change Resistance

Not everyone embraced Scrum immediately. Stakeholders accustomed to making direct requests to developers found the Product Owner intermediary frustrating. Developers who valued autonomy sometimes viewed ceremonies as unnecessary overhead. The organization addressed resistance through demonstrated results. When sprints consistently delivered working software, skeptics recognized the value. Leadership support proved critical. Management allocated time for Scrum ceremonies and protected teams from pressure to bypass the process during urgent situations.

4 Measuring Team Progress and Adaptation

Adopting Agile practices represents only the beginning. Teams must continuously evaluate whether their implementation delivers value and where further improvement is possible. Empiricism requires inspection and adaptation based on transparent information *(Schwaber and Sutherland 2020)*. Research shows that teams demonstrating responsiveness through frequent releases achieve higher stakeholder satisfaction *(Russo, ASE Lecture 3, 2025)*. This chapter examines how Djøf Trade Union measures their Agile effectiveness and evolves their practices.

4.1 Sprint Reviews and Stakeholder Engagement

Djøf Trade Union established regular sprint reviews to evaluate completed work and gather feedback. After each sprint, they review what they accomplished, whether goals were met, and what can be learned **(Transcript: 00:06:48–00:07:31)**. These reviews provide structured moments for reflection rather than pushing forward blindly.

Sprint reviews involve stakeholders from the business side. The team invites interested parties to see what they built, and quarterly demos showcase three months of accumulated progress to broader audiences **(Transcript: 00:20:37–00:21:07)**. This regular demonstration rhythm keeps stakeholders informed without constant interruptions to the development team.

The review process helps the team understand whether their work delivers actual value. When features reach users, the Product Owner and business teams collect feedback and bring it back to development **(Transcript: 00:21:07–00:21:37)**. This creates a feedback loop where user experience influences future prioritization. Success is not just completing tasks but solving real problems for members and internal users.

4.2 Retrospectives and Process Refinement

Beyond reviewing product increments, Ahmed's team holds retrospectives to examine their working methods. These sessions focus on how the team collaborates and what obstacles prevent smooth workflow **(Transcript: 00:07:31–00:07:48)**. Sprint Retrospectives enable teams to reflect on their process and identify improvements for future iterations *(Russo, ASE Lecture 3, 2025)*.

One concrete improvement emerged from retrospective discussions: code reviews were taking too long because colleagues lacked time to examine pull requests promptly **(Transcript: 00:23:07–00:23:37)**. The team agreed that when someone creates a pull request and announces it in Teams, a reviewer must examine it within four hours **(Transcript: 00:23:37–00:24:07)**. This simple commitment dramatically improved their development flow.

The team's pragmatic adaptation of Scrum also appears in retrospectives. When a sprint lacks meaningful progress, they sometimes skip the retrospective because there is little to reflect upon **(Transcript: 00:09:59–00:10:22)**. This flexibility demonstrates their focus on value rather than ceremony for ceremony's sake.

4.3 Tracking Metrics and Leadership Support

Djøf Trade Union tracks concrete indicators of progress. They measure how many tasks complete per sprint, providing insight into team capacity **(Transcript: 00:21:52–00:22:22)**. They also track lead time from when a task starts until it deploys to production, highlighting bottlenecks in their process.

Quality metrics complement velocity tracking. The team monitors how many bugs surface during development versus how many escape to production **(Transcript: 00:22:22–00:22:52)**. This indicates whether their testing catches issues before users encounter them. Quality assurance practices and clear Definition of Done criteria help teams maintain consistent standards *(Russo, ASE Lecture 5, 2025)*. High production bug rates signal problems with testing or code review processes.

The organization's measurement approach remains practical rather than exhaustive. They gather enough data to spot trends and identify problems without drowning in metrics. The goal is actionable insight, not comprehensive dashboards. When code review delays appeared in their data and team discussions, they took concrete action rather than simply documenting the problem.

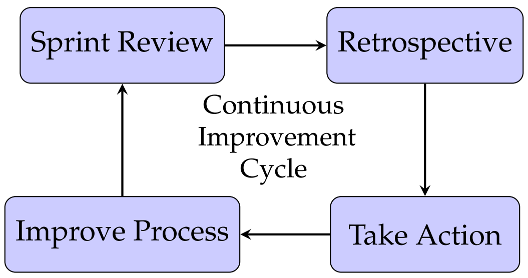

Figure 4.1: *Continuous improvement cycle through regular sprint reviews and retrospectives*

The combination of sprint reviews, retrospectives, and lightweight metrics creates continuous improvement. The organization demonstrates that Agile evaluation doesn't require sophisticated tools or extensive data collection. Regular inspection of both product and process, coupled with willingness to adapt when issues surface, drives meaningful improvement over time.

Management support reinforces this learning orientation **(Transcript: 00:24:22–00:25:22)**, allowing them to invest time in addressing problems rather than only delivering features. Leadership provides resources for skill development and respects process boundaries. When they request tools or training, management evaluates requests seriously. This support creates psychological safety where teams can acknowledge problems without fear, enabling honest retrospectives and genuine improvement.

5

Understanding Stakeholder Voices

Successful Agile implementation depends on understanding and engaging stakeholders effectively. Research with 2,000 Scrum teams found that stakeholder concern is a key driver of team effectiveness *(Verwijs and Russo 2021)*. Djøf Trade Union gradually learned to identify stakeholders and build collaborative relationships that support value delivery.

5.1 Stakeholder Identification and Collaboration

The organization serves two groups: internal employees and union members. Internal employees use systems for booking management and account administration **(Transcript: 00:01:34–00:02:04)**. Union members interact with member-facing systems. These two stakeholder groups create challenges because internal and member needs don't always align.

The Product Owner became the bridge between stakeholder groups and the development team. Before Scrum, tasks arrived chaotically without clear understanding of who needed the work or why. The Product Owner now gathers needs from both groups, translates them into backlog items, and ensures the team understands context **(Transcript: 00:11:22–00:11:52)**.

They recognized that stakeholders have different influence and interest levels. Leadership cares about member growth and operational efficiency. They set budget priorities and decide which initiatives receive funding. Daily system users provide practical feedback about what works and what frustrates them.

Refinement created opportunities for stakeholders to shape solutions before development begins. The team now breaks down epics collaboratively with input from users **(Transcript: 00:05:30–00:06:00)**. When stakeholders participate in defining acceptance criteria, they gain realistic expectations.

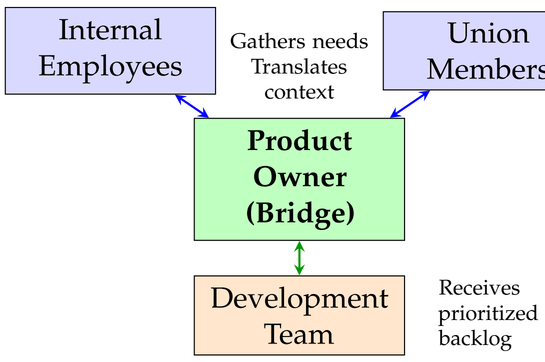

Figure 5.1: *Stakeholder landscape showing two groups served by Ahmed's organization. The Product Owner bridges stakeholder groups and the development team, gathering needs from both internal employees and union members, then translating them into prioritized backlog items.*

Sprint reviews became regular touchpoints where stakeholders see completed work and provide feedback. The team invites business representatives to reviews, demonstrating features and gathering reactions **(Transcript: 00:20:37–00:21:07)**. Quarterly demos expanded this transparency by showcasing three months of progress to broader audiences.

Direct stakeholder-to-developer communication still happens. Ahmed adapted practically: when users approach him with ideas, he listens but redirects them to the Product Owner for prioritization **(Transcript: 00:29:22–00:30:21)**. This protects the team while recognizing valuable ideas.

5.2 Managing Competing Interests and Communication

Stakeholders do not always agree. Economic constraints take priority: when leadership determines something costs too much, the project stops regardless of user enthusiasm **(Transcript: 00:36:17–00:36:47)**. Technical feasibility also matters. When requests would compromise system integrity, developers push back with Product Owner support.

The team learned to find middle ground. Understanding underlying needs often reveals alternative solutions. Successful Agile transformations require balancing multiple influences, with developer skills and management support being critical factors *(Russo 2021)*.

Stakeholder management improved significantly with Agile practices. Structured refinement, regular reviews, transparent backlog, and clear Product Owner ownership created predictability that reduces frustration. Stakeholders know when to expect features. Management sees progress through transparent metrics rather than status reports that may hide problems.

The team established multiple feedback channels tailored to different stakeholder groups. Members receive newsletters announcing new features and changes, creating awareness without direct development team involvement. Internal stakeholders participate in sprint reviews and can observe the Jira backlog. Leadership receives quarterly progress updates aligned with strategic objectives. These layered communication approaches ensure each stakeholder group receives appropriate information without overwhelming the development team with coordination demands.

6

Maintaining Software Quality

Quality became a shared responsibility as Djøf Trade Union adopted Agile practices. Rather than relying on end-of-cycle testing, they built quality checks into every development step. Tests run automatically, code reviews happen before merge, and features deploy to multiple environments.

6.1 Automated Testing and Code Reviews

Djøf Trade Union uses Azure DevOps pipelines to catch problems early. When developers finish a feature, they push code to a branch **(Transcript: 00:12:52–00:13:22)**. The pipeline runs unit tests, integration tests, and end-to-end tests immediately. Failed tests notify developers right away.

Code reviews add another quality layer. A colleague must review every pull request before merge **(Transcript: 00:13:22)**. Their Definition of Done makes quality concrete: code reviewed, tests passing, deployed to staging, and validated **(Transcript: 00:18:37)**.

Bugs still appear. When they do, they fix them immediately without estimation. If too many bugs arrive in one sprint, they reduce feature commitments for the next sprint.

6.2 Legacy Systems and Feedback Integration

Legacy code presents ongoing challenges. Some systems lack documentation, making changes difficult **(Transcript: 00:16:07)**. The team isolates new code from legacy systems where possible.

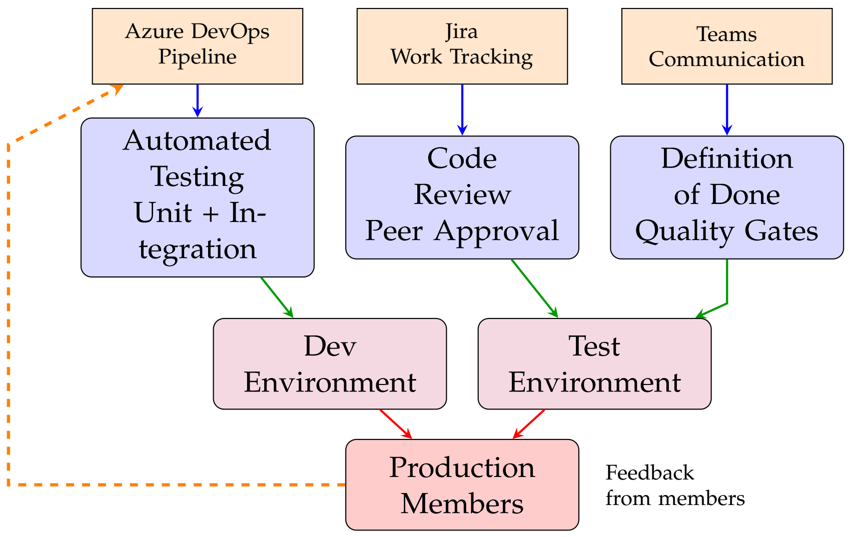

Figure 6.1: *Quality management across tools, practices, and environments. Azure DevOps runs automated tests, Jira tracks work items, and Teams handles communication. Code moves through dev and test environments before reaching production. Member feedback loops back to inform improvements.*

The pipeline improved over time. Manual deployments evolved into automated stages. Developers deploy to dev, then test, then production **(Transcript: 00:13:22)**. Each environment catches different problems.

Azure DevOps manages the build pipeline and testing automation. Teams handles daily communication, and Jira tracks work items **(Transcript: 00:19:22)**.

Sprint reviews and quarterly demos provide quality feedback from stakeholders **(Transcript: 00:20:37–00:21:07)**. The Product Owner gathers member feedback through newsletters and brings it back to the team.

Quality improved through consistent practices. Automated tests, code reviews, multiple deployment stages, and regular feedback work together to catch problems before production.

7

Deployment Practices and System Observability

Djøf Trade Union's deployment pipeline functions reliably but reveals opportunities for improvement. They manually approve each deployment stage, tests sometimes fail unpredictably, and monitoring gaps delay problem detection **(Transcript: 00:37:52)**.

7.1 Current Pipeline and Monitoring Assessment

Their current pipeline moves code through dev, test, and production environments **(Transcript: 00:31:37–00:32:07)**. Automated builds run unit tests, integration tests, and static analysis. However, deployment requires manual approval at each stage **(Transcript: 00:34:07–00:34:37)**.

Tests occasionally fail without reason and require reruns **(Transcript: 00:37:52)**. This instability wastes developer time and erodes confidence in test results. When tests fail randomly, they can't distinguish real problems from false alarms.

Monitoring relies on SMS alerts for critical failures **(Transcript: 00:36:37)**. If problems occur overnight, the team discovers them the next morning. Limited logging makes root cause analysis difficult **(Transcript: 00:37:52–00:38:22)**.

The most impactful improvement would be comprehensive logging and monitoring **(Transcript: 00:38:37)**. Detailed logs capture system behavior. Proactive alerts detect anomalies before they escalate. This reduces response time and prevents user impact.

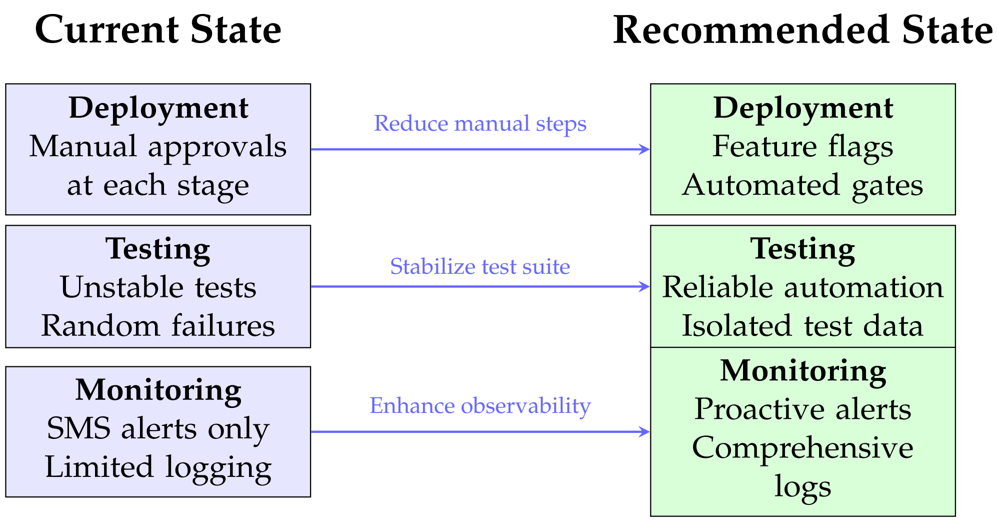

Figure 7.1: *Comparison of current and recommended pipeline practices. Current state relies on manual approvals and reactive monitoring. Recommended improvements focus on automation, test stability, and comprehensive observability to enable faster feedback and reduce operational overhead.*

7.2 Test Stability Improvements

Stabilizing tests eliminates false failures. Isolating test data, managing test dependencies, and identifying flaky tests creates reliable automation. When teams trust their tests, they deploy confidently.

Gradually reducing manual approvals accelerates deployment. Feature flags enable safe production releases without manual gates. Teams can toggle features independently from deployment.

7.3 Building Deployment Capabilities

Enhanced observability transforms operations. Distributed tracing reveals request flows across services. Centralized logging aggregates data for analysis. Custom dashboards surface critical metrics in real time.

Improving deployment frequency builds organizational capability. More frequent deployments reduce batch size, making problems easier to diagnose and fixes faster to deploy.

Investing in test infrastructure pays dividends. Container-based test environments ensure consistency. Parallel test execution reduces feedback time. Regular test suite maintenance removes obsolete tests.

8 Conclusion

Djøf Trade Union transformed chaotic development practices into structured, predictable delivery through pragmatic Scrum adoption. Their journey from ad-hoc task assignment to refined backlog management illustrates how Agile frameworks address fundamental organizational impediments when adapted thoughtfully to context.

The failed membership checkbox project crystallized the costs of absent process controls. Two weeks of development investment vanished because no gatekeeping process verified legal compliance or vendor capability before work began. This painful lesson catalyzed the adoption of refinement sessions that now surface blockers proactively. Their refinement practice demonstrates that breaking down epics collaboratively with clear acceptance criteria prevents wasted effort and aligns stakeholder expectations.

Their flexibility in adapting Scrum reveals maturity beyond rigid framework adherence. Dropping story points when stakeholders misunderstood them and occasionally skipping retrospectives when sprints lacked meaningful progress shows they value outcomes over ceremony. This pragmatism reflects understanding that frameworks serve teams rather than constrain them.

8.1 Emerging Challenges

Legacy system complexity and test instability represent ongoing impediments. Flaky tests that fail randomly erode confidence and waste developer time. Limited monitoring visibility delays problem detection, particularly for overnight failures. These technical gaps suggest areas for continued investment.

8.2 Future Development

The organization built strong foundations through refinement discipline, role clarity, and stakeholder collaboration. Enhancing observability, stabilizing test automation, and gradually reducing manual deployment gates will extend their Agile maturity. Their demonstrated willingness to adapt practices positions them well for continued evolution.

Bibliography

Digital.ai (2024). *State of Agile Report*. Tech. rep. Annual industry survey on Agile adoption and practices. Digital.ai.

Rogers, Philip (Feb. 2024). *Identifying and Engaging Stakeholders*. Medium - A Path Less Taken. Describes usage of the 2x2 Stakeholder Matrix technique for Agile projects. URL: https://medium.com/agile-outside-the-box/identifying-and-engaging-stakeholders-c16c6557317d.

Russo, Daniel (2021). "The Agile Success Model: A Mixed-Methods Study of a Large-Scale Agile Transformation". In: *ACM Transactions on Software Engineering and Methodology*. Study examining success factors for Agile transformations, finding developer skills and management support as most critical factors. URL: https://dl.acm.org/doi/10.1145/3571849.

— (2025a). *ASE Lecture 3: The Scrum Framework*. Course lecture slides. Topics: Scrum roles, events, values, empiricism, responsiveness, stakeholder concern, team effectiveness, continuous improvement, team autonomy.

— (2025b). *ASE Lecture 5: Requirements, Done & Risk Management*. Course lecture slides. Topics: Product Backlog refinement, User Stories, Acceptance Criteria, Definition of Done, risk management.

Sadiq, Ahmed (Fall 2025). *Interview with Ahmed Sadiq - Experienced Software Developer*. Personal interview conducted by Jawad. Trade union organization (internal development department). 10-12 years of experience. Interview conducted in Danish, transcript available in Appendix.

Schwaber, Ken and Jeff Sutherland (Nov. 2020). *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*. Scrum.org. URL: https://scrumguides.org/.

Verwijs, Christiaan, Barry Overeem, and Johannes Schartau (2020). *Scrum: A Framework to Reduce Risk and Deliver Value Sooner*. The Liberators on Medium. Practical guide explaining Scrum framework from empirical process control perspective. URL: https://medium.com/the-liberators/scrum-a-framework-to-reduce-risk-and-deliver-value-sooner-c49fa80902e7.

Verwijs, Christiaan and Daniel Russo (2021). "A Theory of Scrum Team Effectiveness". In: *ACM Transactions on Software Engineering and Methodology*. Research based on data from 1,978 Scrum teams examining factors of team effectiveness. URL: https://dl.acm.org/doi/10.1145/3571849.

Verwijs, Christiaan and Daniel Russo (May 2022). *In-Depth: What Makes Scrum Teams Effective?* The Liberators on Medium. Non-technical version explaining 5 core factors and 13 sub-factors for team effectiveness from research with 2,000 Scrum teams. URL: https://medium.com/the-liberators/what-makes-scrum-teams-effective-7ca9f56a82c7.

Zombie Scrum Resistance (Feb. 2023). *8 Workshops To Involve Your Stakeholders And Deliver More Value Sooner*. The Liberators on Medium. Practical workshops using Liberating Structures for Sprint Reviews, Product Backlog refinement, and stakeholder involvement. URL: https://medium.com/the-liberators/8-workshops-to-involve-your-stakeholders-and-deliver-more-value-sooner-78d07a94efcd.

A

Interview Questions

Note: The following questions were originally asked in Danish. English translations are provided below.

A.1 Welcome and Introduction

1. Hello and thank you for taking the time to meet. My name is Jawad and I study software engineering at AAU Copenhagen. I am working on an exam where I need to interview an experienced developer about how work is conducted as a software developer, and we will spend some time on this. If it's okay with you, I will record the conversation so I can transcribe it afterwards. Is that okay?

A.2 Project Experiences

1. First of all, I would like to talk a bit about project experiences. If you were to show a new team member an example of how things are done at your workplace, which project would you choose that you work with? And why that one specifically?

2. Have you experienced having a project that did not live up to expectations? And if you were to return to that project together with your company, what would you have done differently?

3. When you look back at all the projects and tasks you have been involved in, what characterizes those that just flow and have really good momentum?

4. How do you assess whether your project was a success, not only for your goals within development, but also whether it has delivered value in the real world?

A.3 Work Methods and Daily Practice

1. How would you describe your approach to development? Is there a specific methodology you use, such as Scrum?
2. How has it evolved over time? Have there been changes in the way you have worked within Scrum?
3. Can you tell me about the process from when you receive a task until it reaches the users?
4. How long are your sprints? Who decides what should be done first?
5. What is the best thing about your current way of working? And what is most frustrating?

A.4 The Team and Collaboration

1. Can you paint a picture of the team for me, who is involved, and what are their roles?
2. How do you ensure that everyone is pulling in the same direction?
3. How do you communicate on a daily basis? How much contact do you have with those who actually use the product?

A.5 Measurement and Continuous Improvement

1. How do you keep track of whether you are on the right path? Do you track velocity, lead time, or other metrics?
2. Do you hold retrospectives? Can you give an example of something you have changed based on a retro?
3. Do you feel that management supports your way of working?
4. How do you ensure that competencies in the team stay sharp?

A.6 Quality and Testing

1. When is a task done for you, what needs to be in place before you say finished?
2. What do you do to ensure quality along the way?
3. Can you describe your test approach, which types of tests do you run, and when?
4. What happens when you find a bug, what is the process from when it is discovered until it is fixed?

A.7 AI in Development

1. Have you adopted AI tools for development, Copilot, Claude, Cursor, or similar?
2. What do you primarily use it for? Has it noticeably changed how you work?
3. Has management said anything about what you may and may not do with AI in the codebase?

A.8 CI/CD and Technical Setup

1. Can you describe the journey from when you push code until it runs in production?
2. Which tools do you use in your CI/CD setup?
3. How often do you deploy to production? What happens when something goes wrong?
4. If you had free hands to improve one thing in your setup, what would it be?

A.9 Stakeholders

1. Who do you primarily make the product for? Who are your most important stakeholders?

2. What happens when different stakeholders have opposing interests?
3. How do you keep different groups updated on what is happening in the project?

A.10 Closing

1. How many years have you been doing software development?
2. If you were to give advice to someone who has just started, what is important to focus on?

B

Interview Transcript

The interview was conducted in Danish. Below is the original transcript.

AGILE SOFTWARE ENGINEERING INTERVIEW
AAU KØBENHAVN, AUTUMN 2025

Interviewer: Jawad | *Subject:* Ahmed Sadiq (Senior Developer, 10 to 12 years experience)
Organisation: Trade Union (internal development department)

Velkomst og Introduktion

[00:00:02] Interviewer: Hej tak fordi du ville mødes. Jeg hedder Jawad og jeg studerer software engineering på AAU København. Jeg er i gang med en eksamen hvor jeg skal interviewe en erfaren udvikler om hvordan arbejdet foregår som softwareudvikler, og vi skal bruge lidt tid på det her. Hvis det er okay med dig, optager jeg samtalen, så kan jeg skrive den ud bagefter. Er det i orden?
[Interviewpersonen accepterer]

Sektion 1: Projekterfaringer

[00:00:30] Interviewer: Først og fremmest vil jeg gerne tale lidt om projekterfaringer. Hvis du skal vise et nyt teammedlem et eksempel på hvordan tingene gøres på din arbejdsplads, hvilket projekt vil du vælge som du arbejder med? Og hvorfor lige det?

[00:00:51] Interviewperson: Hvis jeg skulle vise et nyt teammedlem hvordan vi arbejder, så er det jo meget almindelige tools vi bruger. Vi har nogle udviklingsguides liggende, hvor vi fortæller hvordan man sætter sig ind i det og hvilken måde vi arbejder på.
[00:01:05] Interviewperson: Måden vi arbejder på er at vi har et Scrum board i Jira, hvor der ligger en masse user stories som man kan tage på, indtil de er færdige. Et nyt teammedlem vil allerede vide hvilken teknologi han arbejder inden for det vil sige .NET.
[00:01:34] Interviewperson: Det vil sige han skal få sin solution op at køre, og så er det i princippet bare at han går hen og kigger på vores forskellige projekter. Vi har et booking system, vi har et kontosystem og vi har forskellige APIer. Afhængig af hvad den enkelte issue går ud på, så sætter det ham ind i den enkelte løsning.
[00:02:04] Interviewer: Har du oplevet at du har haft et projekt som ikke levede op til forventningerne? Og hvis du skulle tilbage til det projekt sammen med dit firma, hvad har I så gjort anderledes? Har I haft nogle dårlige oplevelser med den måde arbejdet foregår på i jeres projekter?
[00:02:23] Interviewperson: Ja, inden for min tid som ny udvikler fik jeg en opgave. Det er en fagforening jeg arbejder for, og der er det vigtigste nogensinde medlemsoptagelse det vil sige hvordan man får et medlem ind.
[00:02:34] Interviewperson: De ville gerne have at man skulle kunne tilmelde sig en Akasse samtidig. Så jeg skulle lave i et meget gammelt legacy system en lille checkbox, som automatisk kunne sende en mail til Akassen og registrere at brugeren viste interesse i at melde sig ind i kassen.
[00:03:04] Interviewperson: Problemet var bare at i deres medlemssystem kunne vi ikke sikkerhedsindstillingerne tillod ikke at man kunne sætte den indstilling uden at man var logget på som medlem. Og jeg kunne ikke bare lave testmedlemmer.
[00:03:33] Interviewperson: Der var login på devmiljøet, der var login på testmiljøet, men der var ikke login på produktionsmiljøet. Det vil sige jeg kunne sagtens udvikle det, jeg kunne teste det, men jeg kunne ikke deploye til produktion.
[00:04:02] Interviewperson: Det gik tabt efterfølgende. Vi prøvede at presse på for at få fjernet den sikkerhedsindstilling, så vi kunne rette direkte på produktionen. Men så fik vi at vide af den juridiske afdeling at det ikke er tilladt, at vi har den data på den måde.
[00:04:32] Interviewperson: Jeg brugte måske to uger til en måned på den opgave. I virkeligheden skulle den være dræbt meget længe før den overhovedet kom ind til mig. Der manglede en proces opgaven skulle først igennem jura og godkendelse i forhold til GDPR.

[00:04:48] Interviewperson: Og så bagefter skulle den igennem dem der bestemmer omkring sikkerhedsindstillinger på medlemssystemet, som ikke er vores det er en tredjepart inden den overhovedet kom ind til mig. Det var et rigtig dårligt projekt.
[00:05:17] Interviewer: Det giver god mening. Når du kigger tilbage på alle de projekter og opgaver du har været på, hvad kendetegner dem som bare kører og som har en rigtig god flow? Er der nogen principper fra Agile I bruger, eller nogle prioriteringer, noget der gør det nemmere og mere behageligt at arbejde med?
[00:05:30] Interviewperson: Vi har, inden vi går i gang med et projekt, så kører vi altid refinement. Det vil sige at vi har nogle epics som vi sidder og nedbryder til forskellige user stories. Man sidder og nedbryder hele projektet i mange små ting, sådan så vi kan levere features løbende til forretningen i stedet for at komme med det hele på en gang.
[00:06:00] Interviewperson: Inden vi overhovedet går i gang, så har der været en breakdown, og vi skal få behovene ud af forretningen, som vi så nedbryder til vores opgaver. Hvis forretningen har gjort deres arbejde godt nok, og vi går ordentligt til refinement, og hver user story har sit eget klare formål, så er det det der gør at man virkelig lykkes med et projekt og bagefter er i stand til at kunne levere det.
[00:06:24] Interviewer: Er der også andre ting I måler på for eksempel kundetilfredshed? Hvordan vurderer I om jeres projekt var en succes, ikke kun for jeres mål inden for udvikling, men også om det er noget som i den virkelige verden har leveret værdi?
[00:06:48] Interviewperson: Vi kører en gang efter hvert sprint et review, hvor vi som team reviewer hvad vi har lavet i sprintet om vi lykkedes med det, om det var en succes eller en failure, og hvorfor det muligvis ikke lykkedes, så vi kan lære af det.
[00:07:31] Interviewperson: Og så inden i teamet kører vi også nogle retroer, hvor vi prøver at finjustere på de ting vi arbejder med og hvordan vi arbejder med dem.

Sektion 2: Arbejdsmetoder og Daglig Praksis

[00:07:48] Interviewer: Nu vil jeg tale lidt mere om arbejdsmetoder og den daglige praksis. Hvordan vil du beskrive jeres tilgang til udvikling? Er der en bestemt metodologi I bruger, som for eksempel Scrum?
[00:07:50] Interviewperson: Ja, vi bruger Scrum.
[00:07:59] Interviewer: Hvordan har det udviklet sig over tiden? Har der været ændringer i den måde I har arbejdet på inden for Scrum, eller har I bare holdt den fra starten?

[00:08:27] Interviewperson: Vi er ikke helt klassisk Scrum. Vi piller nogle ting fra, fordi vi er en intern udviklingsafdeling. Der er også en masse driftsopgaver som ikke på samme måde kan køre som et projekt. For eksempel fixes du kan ikke køre det som et projekt, du er nødt til at lave det indtil det er færdigt.
[00:08:41] Interviewer: Er der nogle ting I har ændret ved jeres arbejdsmetode med Scrum over tiden?
[00:08:58] Interviewperson: Vi har beholdt de ting som giver mening og droppet de ting som ikke giver mening. Vi har for eksempel droppet story points, fordi det var meget udefinerbart hvad de gik ud på.
[00:09:27] Interviewperson: Det kan godt være at man internt i teamet kunne få en forståelse for hvad de handlede om, men det var meget svært at gøre sådan at kolleger kunne forstå hvad de reelt betød. Vores Scrum master fandt på andre ting hundestørrelser, tshirtstørrelser og alt muligt andet for at gå væk fra tal, fordi når det bare er et tal, så tror man ofte det er dage, og det er det bare ikke.
[00:09:59] Interviewperson: Vi har også nogle gange droppet en retro hvis det ikke gav mening. Vi har droppet et sprint review fordi man ikke rigtig følte det var et sprint man havde kørt i den periode. Så vi bruger Scrummetoderne, men vi tilpasser dem.
[00:10:22] Interviewer: Kan du fortælle mig processen der gennemgås når I har fået en opgave? Hvad sker der fra I har fået opgaven til den når ud til brugerne? Hvilke skridt skal I igennem for at opgaven kan blive ført ud i virkeligheden?
[00:10:52] Interviewperson: Forretningen kommer med en opgave til en product manager. Det er ikke en del af Scrummetoden, det er bare noget for forretningen. Product manageren har et stort Excelark med alle opgaverne.
[00:11:22] Interviewperson: Forretningen vælger så at give opgaven videre til en Product Owner, der prioriterer det i forhold til resten af sine opgaver. Product Owneren tager eventuelle samtaler med forretningen og brugerne for at få behovene ud hvorfor laver vi det her, og hvad prøver I at få løst?
[00:11:52] Interviewperson: Når det er gjort, formulerer de nogle epics. En epic kan i princippet være et helt system. Så er det at vi som team bryder hele projektet ned. Vi sidder og splitter det op i user stories.
[00:12:22] Interviewperson: Så laver vi nogle acceptance criterias hvad er det der skal være opfyldt for at den her feature er færdig? Så kommer den ud på vores backlog. Enten ved sprint start eller undervejs i et sprint tager vi opgaven, og så går vi i gang med at designe det og kode det.
[00:12:52] Interviewperson: Når det er færdigt, skubber vi det op til Azure DevOps hvor der ligger en pipeline. Den kører nogle tests både enhedstests, integrationstests og endtoend tests. Hvis de alle sammen går igennem, så kan vi lave en pull request ind til mainbranchen.
[00:13:22] Interviewperson: En kollega skal så godkende pull requesten der skal

være en review på koden. Når den er godkendt, merges den ind i main. Fra main kan vi så deploye videre til forskellige miljøer først devmiljø, så testmiljø, og til sidst produktionsmiljøet.
[00:13:52] Interviewer: Hvor lange er jeres sprints? Har I eksperimenteret med andre længder?
[00:14:07] Interviewperson: Vi kører tougers sprints. Vi har prøvet tre uger, men det blev for langt. En uge var for kort til at få noget meningsfuldt fra hånden. To uger fungerer godt for os.
[00:14:37] Interviewer: Hvem bestemmer hvad der skal laves først? Hvordan foregår den prioritering?
[00:14:52] Interviewperson: Det er Product Owneren der prioriterer i samarbejde med forretningen. De kigger på business value hvad giver mest værdi for organisationen og medlemmerne. De kigger også på dependencies nogle ting skal være på plads før andre kan laves.
[00:15:22] Interviewer: Hvad er det bedste ved jeres nuværende måde at arbejde på?
[00:15:37] Interviewperson: Flexibiliteten. Vi kan hurtigt tilpasse os hvis prioriteterne ændrer sig. Og så føler jeg vi har god transparens alle ved hvad der foregår og hvorfor.
[00:15:52] Interviewer: Og hvad er mest frustrerende i det daglige arbejde?
[00:16:07] Interviewperson: Nogle gange får vi opgaver der ikke er ordentligt defineret, og så skal vi bruge tid på at finde ud af hvad der egentlig menes. Og så kan legacykoden nogle gange være en udfordring at arbejde med.

Sektion 3: Teamet og Samarbejde

[00:16:37] Interviewer: Kan du tegne et billede af teamet for mig hvem er med, og hvad er deres roller?
[00:16:52] Interviewperson: Vi har en Product Owner der prioriterer opgaver og taler med forretningen. Vi har en Scrum Master der sørger for at Scrumprocesserne bliver fulgt og fjerner blokeringer for teamet.
[00:17:22] Interviewperson: Så har vi udviklingsteamet vi er omkring 5 eller 6 udviklere. Vi har også nogle tech leads der hjælper med de mere tekniske beslutninger og arkitektur.
[00:17:52] Interviewer: Hvordan sikrer I at alle trækker i samme retning at alle ved hvad der er vigtigt lige nu?
[00:18:07] Interviewperson: Vi har sprint goals som bliver kommunikeret ved sprint start. Alle ved hvad målet er for det sprint. Vores backlog er transparent alle kan se hvad der skal laves og i hvilken rækkefølge.
[00:18:37] Interviewperson: Vi har også vores Definition of Done som er klar, så alle ved hvad der skal være på plads før noget er færdigt. Vi holder daily

standups hvor alle fortæller hvad de arbejder på og om der er nogle blokeringer.
[00:19:07] Interviewer: Hvordan kommunikerer I i det daglige standups, Slack, ad hoc møder?
[00:19:22] Interviewperson: Vi har daily standups hver morgen. Vi bruger Teams til daglig kommunikation og hurtige spørgsmål. Vi har også sprint planning, refinement sessions, reviews og retroer som faste møder.
[00:19:52] Interviewperson: Og så tager vi selvfølgelig ad hoc møder når der er behov for det hvis der er et større problem eller noget der skal designes.
[00:20:22] Interviewer: Hvor meget kontakt har I med dem der faktisk bruger produktet?
[00:20:37] Interviewperson: I sprint reviews inviterer vi nogle gange interessenter fra forretningen til at se hvad vi har lavet. Vi har også kvartalsvise demoer hvor vi demonstrerer for en bredere gruppe hvad vi har udviklet over de seneste tre måneder.
[00:21:07] Interviewperson: Med medlemmerne har vi ikke direkte kontakt de får nyhedsbreve når der kommer nye features. Product Owneren og forretningen har mere direkte kontakt med medlemmerne og bringer feedback tilbage til os.

Sektion 4: Måling og Løbende Forbedring

[00:21:37] Interviewer: Hvordan holder I øje med om I er på rette vej? Tracker I velocity, lead time eller andre metrics?
[00:21:52] Interviewperson: Vi tracker hvor mange opgaver vi får færdige per sprint. Vi kigger også på hvor lang tid det tager fra en opgave starter til den er deployed til produktion.
[00:22:22] Interviewperson: Vi har også quality metrics hvor mange bugs finder vi, hvor mange kommer tilbage fra produktion. Det giver os en idé om kvaliteten af vores arbejde.
[00:22:52] Interviewer: Holder I retrospectives? Kan du give et eksempel på noget I har ændret baseret på en retro?
[00:23:07] Interviewperson: Ja, vi holder retro efter hvert sprint. Et eksempel kunne være at vi fandt ud af at vores code reviews tog for lang tid, fordi folk ikke havde tid til at kigge på dem med det samme.
[00:23:37] Interviewperson: Så efter retroen aftalte vi at hvis nogen laver en pull request, så skriver de i Teams, og så skal der være nogen der kigger på den inden for 4 timer. Det gjorde at vores flow blev meget bedre.
[00:24:07] Interviewer: Føler I at ledelsen bakker op om jeres måde at arbejde på? Har I den frihed og de ressourcer I har brug for?
[00:24:22] Interviewperson: Ja, overordnet set får vi god opbakning. Ledelsen forstår at vi har brug for tid til at gøre tingene ordentligt. Hvis vi siger at noget tager længere tid end forventet, så lytter de.

[00:24:52] Interviewperson: De prøver også at fjerne blokeringer for os hvis vi mangler adgang til noget eller skal have godkendelser fra andre afdelinger, så hjælper de med det.
[00:25:22] Interviewer: Hvor gode er I til at skifte retning når kravene ændrer sig?
[00:25:37] Interviewperson: Det er en af fordelene ved Agile vi kan forholdsvis nemt ændre kurs. Hvis forretningen kommer og siger at prioriteterne har ændret sig, så kan vi tage det op til næste sprint planning og ændre hvad vi arbejder på.
[00:26:07] Interviewperson: Nogle gange sker det også midt i et sprint, men så prøver vi at undgå det, medmindre det er virkelig kritisk.
[00:26:37] Interviewer: Hvordan sørger I for at holde kompetencerne skarpe i teamet?
[00:26:52] Interviewperson: Vi har mulighed for at tage kurser og certificeringer. Vi deler også viden internt hvis nogen har lært noget nyt, så holder de en kort præsentation for resten af teamet. Og så lærer vi meget af hinanden gennem code reviews og pair programming.

Sektion 5: Kvalitet og Test

[00:27:22] Interviewer: Hvornår er en opgave done for jer hvad skal være på plads før I siger færdig?
[00:27:37] Interviewperson: Vi har en Definition of Done. Den siger at: Koden skal være skrevet og opfylde acceptance criteria. Der skal være unit tests der dækker den nye kode. Koden skal være code reviewed og godkendt. Integration tests skal køre uden fejl. Det skal være deployed til testmiljø og testet der. Dokumentation skal være opdateret hvis relevant. Product Owner skal have accepteret det.
[00:28:22] Interviewer: Hvad gør I for at sikre kvaliteten undervejs code review, pair programming, static analysis?
[00:28:37] Interviewperson: Vi bruger code reviews på alle pull requests mindst én kollega skal godkende før kode merges. Vi bruger ikke så meget pair programming, men nogle gange hvis noget er særligt komplekst eller hvis en junior udvikler skal lære noget.
[00:29:07] Interviewperson: Vi har også static analysis der kører automatisk i vores pipeline det tjekker for code style issues og potentielle bugs.
[00:29:37] Interviewer: Kan du beskrive jeres test approach hvilke typer tests kører I, og hvornår?
[00:29:52] Interviewperson: Vi har flere lag af tests. Unit tests de kører lokalt når vi udvikler og i pipelinen når vi pusher kode. De tester individuelle funktioner og metoder.
[00:30:22] Interviewperson: Integration tests de kører i pipelinen og tester at

forskellige dele af systemet fungerer sammen. Endtoend tests de kører også i pipelinen og simulerer rigtige brugerflows igennem hele systemet.
[00:30:52] Interviewperson: Nogle gange laver vi også manuel testing på testmiljøet før vi deployer til produktion.
[00:31:22] Interviewer: Hvor meget af jeres testing er automatiseret? Har I tillid til at tests fanger problemerne?
[00:31:37] Interviewperson: De fleste af vores tests er automatiseret. Vi har god code coverage med vores unit tests. Men vi stoler ikke 100 procent på dem der kan altid være edge cases vi ikke har fanget.
[00:32:07] Interviewperson: Så vi laver også manuel testing af nye features inden de går til produktion. Men automatiseringen giver os god tillid til at vi ikke har ødelagt noget eksisterende når vi laver ændringer.
[00:32:37] Interviewer: Hvad sker der når I finder en bug hvad er processen fra den opdages til den er fixet?
[00:32:52] Interviewperson: Hvis vi finder en bug i test, så opretter vi en bug ticket i Jira. Den bliver prioriteret af Product Owner hvis det er kritisk, får den høj prioritet og vi fixer den med det samme. Hvis det er mindre vigtigt, kommer den i backloggen.
[00:33:22] Interviewperson: Hvis der kommer en bug fra produktion, så evaluerer vi hvor alvorlig den er. Hvis det er noget der stopper brugerne, så dropper vi hvad vi har i hænderne og fixer det. Ellers kommer den i backloggen som en prioriteret opgave.

Sektion 6: AI i Udviklingen

[00:33:52] Interviewer: Har I taget AIværktøjer i brug til udvikling Copilot, Claude, Cursor eller lignende?
[00:34:07] Interviewperson: Ja, vi bruger GitHub Copilot.
[00:34:22] Interviewer: Hvad bruger I det primært til? Hvor giver det mest værdi i hverdagen?
[00:34:37] Interviewperson: Vi bruger det hovedsageligt til code completion når man skriver kode, så foreslår den hvordan resten af funktionen kunne se ud. Vi bruger det til at skrive boilerplate kode og repetitiv kode hurtigere.
[00:35:07] Interviewperson: Det er faktisk ret godt til at skrive unit tests den kan generere test cases. Og nogle gange bruger vi det til at forklare kompleks kode man ikke helt forstår.
[00:35:37] Interviewer: Har det ændret noget mærkbart ved hvordan I arbejder?
[00:35:52] Interviewperson: Ja, det har gjort os hurtigere til visse opgaver. Især den repetitive kode går meget hurtigere. Men man skal stadig vide hvad man laver AIen foreslår ikke altid den bedste løsning, så man skal kunne vurdere om det den foreslår giver mening.

[00:36:22] Interviewer: Har I oplevet situationer hvor AIgenereret kode har skabt problemer?
[00:36:37] Interviewperson: Vi har heldigvis ikke oplevet større problemer endnu, men man skal være opmærksom. Koden er ikke altid den bedste. Nogle gange foreslår den løsninger der virker, men som ikke følger vores coding standards eller best practices. Så man skal stadig review koden ordentligt, selvom den kommer fra AI.
[00:37:07] Interviewer: Har ledelsen sagt noget om hvad I må og ikke må med AI i kodebasen?
[00:37:22] Interviewperson: Vi har haft nogle diskussioner om det. Hovedreglen er at man stadig er ansvarlig for den kode man skriver, uanset om det er AIgenereret eller ej. Og man skal være forsigtig med ikke at sende følsom forretningskode til eksterne AItjenester.
[00:37:52] Interviewer: Hvordan ser du AI i fremtiden for jeres arbejdsplads?
[00:38:07] Interviewperson: Jeg tror AI vil gøre at vi bruger færre udviklere i fremtiden. Men jeg tror ikke det erstatter os helt. Der vil stadig være behov for nogen der forstår forretningen, kan designe løsninger og vurdere om den kode AI genererer faktisk løser problemet ordentligt. AI er et værktøj der gør os mere produktive, men det kræver stadig erfarne udviklere til at guide det og sikre kvaliteten.

Sektion 7: CI/CD og Teknisk Setup

[00:38:37] Interviewer: Kan du beskrive rejsen fra du pusher kode til det kører i produktion hvilke trin passerer det igennem?
[00:38:52] Interviewperson: Når jeg pusher kode til Azure DevOps, så bygges koden automatisk. Unit tests kører, integration tests kører, og static analysis kører.
[00:39:22] Interviewperson: Hvis alt går igennem, kan jeg lave en pull request til main branch. En kollega skal code reviewe og godkende. Når den er godkendt, merges den til main. Fra main kan vi deploye til devmiljø først, så til testmiljø hvor vi tester manuelt, og til sidst til produktion når alt ser godt ud.
[00:39:52] Interviewer: Hvilke tools bruger I i jeres CI/CD setup GitHub Actions, Jenkins, Azure DevOps eller noget andet?
[00:40:07] Interviewperson: Vi bruger Azure DevOps til det hele. Det er vores version control system, vores CI/CD platform, der kører vores pipelines og builds. Vi bruger Azure Containers til at køre vores applikationer i skyen. Og vi har logging i Azure portalen, så vi kan se hvad der sker i vores systemer. Det er smart at have det hele i én platform det gør det nemmere at arbejde med.
[00:40:37] Interviewer: Hvor meget kører automatisk vs. hvor meget kræver manuel godkendelse?

[00:40:52] Interviewperson: Build og test er fuldt automatisk det starter hver gang nogen pusher kode. Men deployment er delvist manuelt. Vi skal godkende at koden deployes til hvert miljø. Jeg kan ikke bare pushe kode direkte til produktion. Det skal igennem en branch, lave en pull request, få code review, blive godkendt, og så kan vi deploye. Det giver os kontrol og sikkerhed for at forkert kode ikke kommer direkte ud til brugerne.
[00:41:22] Interviewer: Hvor ofte deployer I til produktion? Er det kontinuerligt, dagligt, ugentligt, eller efter en specifik sprint?
[00:41:37] Interviewperson: Det varierer meget. Hvis der er noget der bliver færdigt i løbet af et sprint, så bliver det deployed. Når en feature er færdig, skal den selvfølgelig deployes. Vi deployer ofte til udviklings og testmiljøer. Til produktion er det som minimum en gang per sprint cirka, men det kan godt være oftere hvis vi har flere features færdige.
[00:42:07] Interviewer: Hvad sker der når noget går galt i produktion hvem får alarm, og hvad er processen?
[00:42:22] Interviewperson: Vi har SMSovervågning sat op. Det vil sige at hvis der er noget kritisk der går galt, får vi SMSalarmer. Hvis det sker om natten, må vi håbe at nogen våger. Ellers opdager vi det om morgenen.
[00:42:52] Interviewperson: Vi har altid en tidligere version liggende klar. Hvis der går noget alvorligt galt, kan vi rulle tilbage til den forrige løsning hurtigt. Det tager typisk kun få minutter at gå tilbage til en version der virker. Så kan vi i løbet af dagen analysere hvad der gik galt og lave et ordentligt fix.
[00:43:22] Interviewer: Hvor oplever I de største udfordringer i jeres pipeline?
[00:43:37] Interviewperson: Nogle gange kan vores tests være ustabile de fejler af og til uden grund og skal bare køres igen. Det kan være frustrerende. Og så kunne vores monitoring være bedre. Vi kunne logge meget mere og have flere alarmer sat op, så vi hurtigere opdager når noget går galt.
[00:44:07] Interviewer: Hvis du fik frie hænder til at forbedre én ting i jeres setup, hvad ville det være?
[00:44:22] Interviewperson: Jeg ville forbedre vores logging og monitoring betydeligt. Vi kunne logge meget mere detaljeret information, og vi kunne sætte langt flere alarmer op, så vi proaktivt får besked når noget ser mærkeligt ud i stedet for at vente til systemet går helt ned.

Sektion 8: Stakeholders og Omverdenen

[00:44:52] Interviewer: Hvem er det I primært laver produktet for? Hvem er jeres vigtigste interessenter?
[00:45:07] Interviewperson: Det afhænger af hvilket system vi arbejder med. Nogle gange er det internt i forretningen altså vores kollegaer der bruger systemerne. Men medlemmerne er helt klart de primære brugere. De er dem vi

ultimativt laver det for. Og så er der selvfølgelig også forretningen og ledelsen som stakeholders.
[00:45:37] Interviewer: Hvad sker der når forskellige stakeholders har modsatrettede interesser?
[00:45:52] Interviewperson: Økonomi er helt klart vigtigt. Hvis organisationen siger at noget er for dyrt eller ikke kan lade sig gøre økonomisk, så stopper vi projektet. Hvis forretningsønsker ikke giver mening i forhold til systemet teknisk, så er det os der vinder. Det kan godt være de vil have en knap der blinker, men hvis den knap ødelægger hele systemet, så får de den ikke. Men generelt prøver vi at finde kompromiser hvor alle parter kan få noget af det de vil have, og hvor vi leverer værdi til medlemmerne.
[00:46:22] Interviewer: Hvordan holder I forskellige grupper opdateret på hvad der sker i projektet?
[00:46:37] Interviewperson: Internt holder vi kvartalsvise demoer, hvor vi demonstrerer hvad vi har lavet i de seneste tre måneder for forretningen og ledelsen. Medlemmerne får nyhedsbreve når der kommer nye features eller ændringer i systemerne. Vi har ikke direkte kommunikation med medlemmerne det håndteres af andre afdelinger der så bringer feedback til os.

Afrunding

[00:47:07] Interviewer: Hvor mange år har du været i gang med softwareudvikling?
[00:47:12] Interviewperson: Det er omkring 10 til 12 år.
[00:47:22] Interviewer: Hvis du skulle give et råd til nogen som lige er startet hvad er det vigtige man skal fokusere på for at få et godt arbejdsmiljø og kunne fungere godt og levere noget?
[00:47:37] Interviewperson: Udover at man selvfølgelig skal være god til at løse problemer, så er det måske også en meget god idé at lære at forklare hvad man laver. Det kan være svært nogle gange at forklare tekniske ting til ikketekniske personer, men det er super vigtigt. Lærer man at kommunikere godt, både med kollegaer og med forretningen, så kommer man meget langt.
[00:48:07] Interviewer: Mange tak for din tid. Hvis det er i orden, må jeg gerne skrive til dig hvis der dukker nogle opfølgende spørgsmål op?
Interviewperson: Selvfølgelig, det er helt i orden.

Slut på interview

C

Thematic Analysis of Interview Data

This appendix documents the thematic analysis process applied to Ahmed's interview responses. The analysis followed an inductive approach, allowing themes to emerge organically from the data.

C.1 Familiarization and Initial Observations

During transcript review, several patterns emerged. Ahmed spoke extensively about process adaptation, describing how the team modified standard Scrum practices to fit their organizational context. His narrative about the failed legacy system task revealed the importance of proper requirements validation. The discussion of refinement sessions highlighted how structured preparation prevents problems.

C.2 Coding Process

The transcript was systematically coded, identifying meaningful segments:

Pragmatic Scrum Adaptation
Team dropped story points because stakeholders misinterpreted them as time estimates. Scrum Master experimented with dog sizes and t shirt sizes. They occasionally skip ceremonies when sprints lack meaningful progress (00:08:58).

Refinement as Prevention
Structured refinement sessions decompose epics into user stories with clear acceptance criteria. When refinement is thorough and business requirements are well articulated, projects succeed (00:05:30).

Process Gate Failures
The legacy checkbox task failed because it bypassed legal review and third party coordination. Ahmed spent weeks on work that should have been blocked earlier (00:02:23).

Definition of Done Discipline
Comprehensive DoD: code must pass review, unit tests must cover new code, integration tests must pass, staging deployment and testing must complete, documentation updated, PO acceptance (00:27:37).

Flexible Sprint Boundaries
Two week sprints provide structure, but team can accommodate critical changes mid sprint. Regular sprint planning allows course correction (00:25:37).

Pipeline Quality Gates
Azure DevOps pipelines automatically run unit tests, integration tests, and end to end tests. Code cannot merge without passing all gates and receiving colleague approval (00:12:52).

AI Tool Integration
GitHub Copilot accelerates boilerplate and repetitive code. Developers must still review AI suggestions against coding standards. Tool helps but does not replace judgment (00:34:37).

Stakeholder Complexity
Multiple stakeholder groups: internal business users, external members, management. Economic constraints often override feature requests. Technical feasibility can veto business wishes (00:45:07).

Monitoring Limitations
Current monitoring relies on SMS alerts for critical failures. Improved logging and proactive alerting identified as primary enhancement area (00:43:37).

Communication Skills
Beyond technical ability, explaining technical concepts to non technical stakeholders is crucial for career success (00:47:37).

C.3 Theme Construction

Codes were organized into four major themes through iterative grouping:

Theme A: Adaptive Framework Implementation. The team treats Scrum as a starting point rather than a rigid prescription. When estimation caused confusion, they experimented with alternatives and moved away from numerical estimates. Their willingness to skip ceremonies when they would not add value demonstrates a focus on outcomes over rituals. *Key evidence:* "Vi har beholdt de ting som giver mening og droppet de ting som ikke giver mening" (00:08:58).

Theme B: Process as Risk Mitigation. The failed legacy task fundamentally shaped Ahmed's understanding of why process gates exist. The checkbox feature seemed simple but required legal approval, third party coordination, and production environment access. Today, the team's refinement process explicitly addresses such dependencies. *Key evidence:* "I virkeligheden skulle den være dræbt meget længe før den overhovedet kom ind til mig" (00:04:32).

Theme C: Balancing Autonomy and Control. The deployment pipeline provides automatic quality enforcement through tests and code review requirements. This creates control without requiring manual intervention for routine situations. However, deployment to production requires explicit approval, maintaining human oversight at critical transitions. *Key evidence:* "Jeg kan ikke bare pushe kode direkte til produktion" (00:40:52).

Theme D: Continuous Improvement Through Reflection. Sprint reviews evaluate what was delivered and whether it met expectations. Retrospectives examine how the team worked and identify process improvements. The code review response time improvement illustrates the retrospective process in action. *Key evidence:* "Så efter retroen aftalte vi at hvis nogen laver en pull request... skal der være nogen der kigger på den inden for 4 timer" (00:23:37).

C.4 Theme Relationships

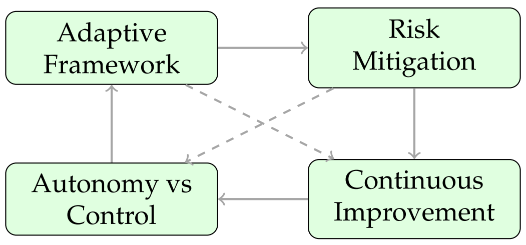

Figure C.1: *Theme interconnections in Ahmed's interview*

The four themes interconnect significantly. Adaptive Framework Implementation enables Process as Risk Mitigation because the team can modify practices based on what they learn from failures. Balancing Autonomy and Control creates the structure within which Continuous Improvement operates. The retrospective process feeds insights back into framework adaptation.

C.5 Summary

The analysis reveals a trade union development department that has internalized Agile principles while maintaining practical flexibility. Rather than following Scrum literally, they extract value from its practices while adapting to their context as an internal team serving multiple stakeholder groups. Ahmed's perspective, shaped by over a decade of experience including early failures, emphasizes process discipline not as bureaucracy but as protection against avoidable problems.