DOI 10.54596/2958-0048-2025-2-184-193
UDK 004.4
IRSTI 28.23.29

AGILE METHODOLOGIES IN BANKING SOFTWARE: A CASE-BASED ANALYSIS OF IMPLEMENTATION AT SMART SOLUTION

Kulikova E.V.[1*]
[1*]*Smart Solution, located Aurora, Ontario, Canada*
[*]*Corresponding author: ekulikova@smartsolution.com*

Abstract

The implementation of Agile frameworks in banking software development has gained momentum in recent years, driven by the need for flexibility, faster delivery, and greater responsiveness to change. This paper presents a case study of Smart Solution (hereafter referred to as *Smart*), a Canadian banking software provider, and explores the practical application of Scrum, Kanban, and Scrumban methodologies within a regulated financial environment. Through an analysis of internal practices, team structure, tooling, and project types, the study examines how Agile principles have been adapted to meet compliance standards, maintain predictable release cycles, and balance multiple parallel workstreams. Challenges encountered include tool misalignment, inconsistent sprint focus, and the limitations of Scrum for maintenance work. Findings suggest that Scrumban offers a viable hybrid solution, combining structured planning with continuous flow. The paper concludes with recommendations for implementing Agile in the banking domain and proposes several hypotheses for future research.

Keywords: Agile, Scrum, Kanban, Scrumban, Banking Software, Fintech, Compliance, Software Development, Case Study.

БАНКТЫҚ БАҒДАРЛАМАЛЫҚ ҚАМТАМАСЫЗ ЕТУДЕ AGILE ӘДІСТЕМЕЛЕРІН ЕНГІЗУ: SMART SOLUTION КОМПАНИЯСЫНЫҢ МЫСАЛЫНА НЕГІЗДЕЛГЕН ТАЛДАУ

Куликова Е.В.[1*]
[1*]*Smart Solution, Аврора, Онтарио, Канада*
[*]*Хат-хабар үшін автор: ekulikova@smartsolution.com*

Аңдатпа

Соңғы жылдары банктық бағдарламалық қамтамасыз етуді әзірлеу саласында Agile-фреймворктерді енгізу айтарлықтай қарқын алды. Бұл үрдіс икемділікке, шешімдерді жедел жеткізуге және өзгерістерге жедел бейімделуге деген сұраныстың артуымен түсіндіріледі. Осы мақалада Канаданың банк саласына арналған IT-шешімдер жеткізушісі – Smart Solution (бұдан әрі – Smart) компаниясының мысалында Scrum, Kanban және Scrumban әдістемелерін реттелетін қаржылық ортада қолданудың тәжірибелік қырлары қарастырылады. Ішкі процестер, команда құрылымы, қолданылатын құралдар және жобалардың түрлері бойынша жүргізілген талдау негізінде Agile қағидаттарының сәйкестік талаптарын орындау, шығарылым циклдарының болжамдылығын сақтау және бірнеше параллель жұмыс ағындарын тиімді үйлестіру үшін қалай бейімделгені сараланады. Туындаған негізгі қиындықтар қатарына құралдардың сәйкес келмеуі, спринттердегі фокус тұрақсыздығы және техникалық қолдау жұмыстары үшін Scrum әдісінің шектеулері жатады. Зерттеу нәтижелері Scrumban әдістемесінің құрылымдалған жоспарлауды үздіксіз ағыммен үйлестіретін тиімді гибридтік шешім бола алатынын көрсетеді. Мақалада банк саласында Agile әдістерін енгізу бойынша ұсыныстар беріліп, болашақ зерттеулерге арналған бірнеше гипотеза ұсынылады.

Кілт сөздер: Agile, Scrum, Kanban, Scrumban, банктық бағдарламалық қамтамасыз ету, финтех, сәйкестік, бағдарламалық жасақтама әзірлеу, кейстік талдау.

МЕТОДОЛОГИЯ AGILE В БАНКОВСКОМ ПРОГРАММНОМ ОБЕСПЕЧЕНИИ: АНАЛИЗ ВНЕДРЕНИЯ НА ПРИМЕРЕ КОМПАНИИ SMART SOLUTION

Куликова Е.В.[1*]

[1*]*Smart Solution, Аврора, Онтарио, Канада*

[*]*Автор для корреспонденции: ekulikova@smartsolution.com*

Аннотация

В последние годы внедрение фреймворков Agile в разработку банковского программного обеспечения значительно активизировалось, что обусловлено необходимостью повышения гибкости, ускорения поставки решений и повышения адаптивности к изменениям. В настоящей статье представлен кейс Smart Solution (далее – Smart), канадского поставщика банковских ИТ-решений, с акцентом на практическое применение методологий Scrum, Kanban и Scrumban в условиях нормативно регулируемой финансовой среды. На основе анализа внутренних процессов, структуры команды, используемых инструментов и типов проектов рассмотрено, каким образом принципы Agile были адаптированы для соблюдения требований соответствия, обеспечения предсказуемости релизов и эффективного управления параллельными потоками задач. Среди выявленных проблем – несоответствие инструментов, отсутствие устойчивого фокуса в спринтах и ограничения Scrum при выполнении задач технической поддержки. Полученные результаты свидетельствуют о том, что Scrumban представляет собой жизнеспособную гибридную модель, сочетающую структурированное планирование с гибкостью непрерывного потока. В завершение представлены рекомендации по применению Agile в банковской сфере и сформулированы гипотезы для дальнейших исследований.

Ключевые слова: Agile, Scrum, Kanban, Scrumban, банковское программное обеспечение, финтех, соответствие требованиям, разработка ПО, кейс-стади.

1. Introduction

In today's fast-evolving landscape of digital innovation and AI integration, the need for adaptable, collaborative, and responsive software development in banking is more critical than ever. Technologies like generative AI, predictive analytics, and intelligent automation are reshaping customer expectations and operational possibilities across the financial sector [9]. Implementing a well-structured Agile approach not only accelerates delivery but also provides the frameworks and discipline necessary to safely and effectively adopt AI-driven tools and features. This aligns with foundational Agile principles emphasizing adaptive delivery and iterative value creation [3].

Agile methodologies empower development teams to experiment, validate, and iterate on AI capabilities within controlled environments – ensuring quality, compliance, and user value. At the same time, it's important to recognize that even fintech firms – despite their innovative edge – tend to adopt major changes like AI cautiously and incrementally, due to regulatory, security, and operational constraints. This cautious, staged approach is consistent with how financial services have historically adopted Agile methodologies [6].

While AI serves as a powerful backdrop to today's transformation in financial services, this article focuses specifically on the implementation of Agile methodologies in the banking software development process. The discussion uses the case of Smart, a Canadian banking software provider, to explore real-world challenges, adjustments, and lessons learned in applying Agile practices. Though the intersection of Agile and AI adoption is increasingly relevant, the goal here is not to examine AI use cases directly, but rather to understand how Agile foundations – when properly established – can support and accelerate broader digital innovation, including AI, in a secure and sustainable way.

2. Background and Literature Review

Agile methodologies originated as a response to the inefficiencies of traditional Waterfall approaches, which emphasized comprehensive upfront planning and sequential delivery. The

2001 Agile Manifesto introduced values such as individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan [3].

Among the most widely adopted Agile frameworks are Scrum, Kanban, and Scrumban. Each of these approaches offers distinct principles, roles, and workflows:

- Scrum operates in fixed-length sprints (typically 2–4 weeks) and is driven by structured ceremonies such as sprint planning, daily standups, reviews, and retrospectives [7].
- Kanban is flow-based, with work continuously pulled through a system visualized on a board. It emphasizes limiting work in progress (WIP) and optimizing cycle times [1].
- Scrumban is a hybrid model that blends the structure of Scrum with the flexibility of Kanban, making it especially suitable for teams with mixed types of work and shifting priorities [5].

In regulated industries such as banking, Agile practices face additional scrutiny. Documentation, auditability, and compliance processes cannot be deprioritized in favor of speed. As a result, Agile adaptations in financial environments often incorporate elements of traditional models, forming hybrid workflows that balance agility with regulatory compliance [6].

Literature on Agile in banking contexts [3, 6] highlights the need for customized Agile approaches that address domain-specific risks and challenges, such as siloed teams, strict change management processes, and long-term vendor contracts.

3. Research Context and Methodology

This study is grounded in a qualitative case study of Smart Solution, a Canadian fintech software company specializing in core banking solutions for credit unions and financial institutions. The company's software products manage critical functions such as loan origination, deposits, account servicing, reporting, and compliance support.

Smart has undergone a multi-year transformation in its approach to software development, evolving from traditional Waterfall practices toward a hybrid Agile model centered on Scrumban. The study draws upon internal documentation, meeting notes, tool configurations, sprint records, and team retrospectives to reconstruct the organization’s Agile journey [9].

Key characteristics of the research setting include:

- A cross-functional team of ~28 practitioners (BAs, developers, QA, product leads).
- A bi-monthly release cadence affecting all product lines.
- Simultaneous execution of multiple project streams: new features, client implementations, regulatory updates, and ongoing support.
- Historical use of tools such as Eventum (for ticket tracking), Jira (for Agile workflows), and Confluence (for documentation).

Data collection focused on four dimensions:

1. Process Design: Sprint structure, Kanban board usage, backlog management.
2. Team Behavior: Meeting cadence, role adherence, cross-functional collaboration.
3. Project Type: Feature development vs. support/maintenance.
4. Outcomes: Sprint completion rates, perceived project clarity, ability to meet release deadlines.

The goal of this methodology is not to produce statistically generalizable findings, but to provide a practice-based lens through which Agile adaptation in banking software can be better understood, tested, and refined.

4. Implementation Findings

The following findings are derived from Smart's sprint tracking data, stakeholder feedback, tool usage records, and internal retrospectives, as described in the methodology section.

The initial adoption at Smart Solution faced significant organizational resistance, particularly from leadership. The development manager at the time was skeptical of Agile practices and Jira as a tool. Jira was introduced without a clear implementation plan and was configured by a developer who lacked familiarity with Agile principles, Scrum ceremonies, and Smart's operational context. As a result, the rollout failed. Teams became frustrated with confusing workflows and lack of transparency, reinforcing resistance to Agile rather than encouraging adoption. As shown in Table 1**,** stakeholder sentiment toward Agile tools was initially mixed.

Table 1. Stakeholder Sentiment Toward Agile Tools at Smart (Initial Phase)

Role	Supporter (%)	Neutral (%)	Opposed (%)
Development Manager	0%	75%	25%
Business Analyst	25%	50%	25%
Developer	50%	50%	0%
QA Analyst	50%	50%	0%

It took several years before Smart revisited Jira and Agile practices seriously. This time, a developer experienced in both Scrum and Jira administration led the implementation. A small project team was again created as a controlled pilot, and Jira was configured to reflect real team workflows. This more thoughtful and technically sound setup led to broader team engagement and confidence in the system (Smart Solution internal documentation, 2017–2024). As shown in Table 2**,** second-phase implementation saw improved stakeholder attitudes.

Table 2. Stakeholder Sentiment Toward Agile Tools at Smart (Second Phase)

Role	Supporter (%)	Neutral (%)	Opposed (%)
Development Manager	50%	50%	0%
Business Analyst	50%	50%	0%
Developer	75%	25%	0%
QA Analyst	75%	25%	0%

Note: Tables above are calculated based on all development-involved employees in Smart Solution, as all of them got access to Jira and other tools, even though a pilot project started with five (5) people.

Smart's Agile transformation began with a focused pilot using Scrum. A dedicated team – including a Product Manager acting as Product Owner, a BA, two developers (one of them also acting as the Scrum Master), and a QA analyst – was assigned a greenfield project with minimal outside distractions. Using Jira for backlog and sprint tracking, the team followed structured Scrum ceremonies. The project was delivered successfully without a scope creep, albeit with unanticipated overhead in sprint planning, daily meetings, and other Scrum ceremonies. This experience confirmed the benefits of Agile's structure but highlighted the time burden on small, multi-role teams. As indicated in Table 3**,** actual hours significantly exceeded estimated hours for the pilot.

Table 3. Time Estimated vs Actual Time Spent by roles on a pilot project

Role	Estimated (hrs)	Actual (hrs)
Product Owner	20	80
Business Analyst	45	100
Developer	120	250
QA Analyst	100	210
Total	285	640 (~225% over)

Table 4 compares actual and estimated hours across two subsequent projects.

Table 4. Time Estimated vs Actual Time Spent by roles on the next two projects (same team, primarily dedicated to the Scrum projects (~85% of their time))

Role	Project 2 Estimated (hrs)	Project 2 Actual (hrs)	Project 3 Estimated (hrs)	Project 3 Actual (hrs)
Product Owner	40	60	50	60
Business Analyst	80	100	80	90
Developer	300	375	450	500
QA Analyst	220	300	300	340
Total	640	835 (~130% over)	880	9

The accuracy of estimates was significantly improved (225% of initial estimates vs 113% of initial estimates) as the team got better at estimates, and after a few sprints, the velocity stabilized around 65-70 points per two-week sprints, which in turn helped with sprint planning and load distribution. However, the cost of projects went up as the team allocated more time on daily Scrum meetings and other Scrum ceremonies, which introduced one of the drawbacks.

Following the success of Scrum implementation for the dedicated team, Smart expanded Scrum to additional teams. Results were mixed. Scrum worked reasonably well for new development when teams were stable, but struggled when developers, BAs, and QAs were split across multiple projects. Shared resource constraints and mid-sprint task switching often led to broken sprint commitments.

The size of the departments (BA, Development and QA) at Smart Solution and the complexity of the products being delivered didn't allow to keep people dedicated to one project only; and a senior Subject Matter Expert (SME) would be required to participate in 3 or 4 projects at the same time, with maintenance tickets on top of the projects load.

That lead to significant drop in velocity per team per project, scripts with accomplishments way below the projected velocity, and overall teams' frustration.

Table 5 demonstrates the impact of team allocation on sprint performance.

Table 5. Scrum Performance by Resource Allocation Type

Team Allocation Type	% Sprints Completed with 90% of planned velocity (or higher)	Lowest % of points completed in Sprint vs planned
Dedicated to Project	90%	70%
Split Between Projects	50%	25%

The use of Scrum for maintenance work (e.g., bug fixes, small enhancements) proved ineffective. Maintenance tasks arrived unpredictably, required quick turnaround, and frequently

bypassed the sprint cadence. Attempting to force them into sprint cycles led to frequent scope disruptions, misaligned priorities, and low team morale.

Also, as all members of all teams might be pulled into working on maintenance/support items depending on their area of expertise, creation of a big Scrum "maintenance" project quickly proved as non-efficient: having all 28 members be present in daily Scrum meetings bloated meeting time up without adding efficiency in resolving issues.

Table 6 shows reduced meeting efficiency in the maintenance Scrum setup.

Table 6. Meeting Efficiency in Scrum for Maintenance

Metric	Scrum New Feature Project	Scrum Maintenance Project
Avg. Daily Meeting Duration	30 min	45 min
Avg. Attendance (team members)	5-6	25+
Avg. Items Resolved per Meeting	6-8	1–2

Meeting time and attendance were calculated based on the employees' time tracking during one month of Scrum Maintenance project try out.

As a response, Smart moved toward using Kanban (via Eventum) for handling maintenance requests, client-specific small customizations, and support tasks. This flow allowed more responsive task intake without formal iteration planning [9].

As shown in Figure 1, the volume and variability of tickets were significant. The chart below depicts the number of maintenance tickets per two-month release cycle over the past eight years, ranging from 25 (lowest, February 2020 release) to 115 (highest, September 2018 release). These figures exclude support tickets that did not require code changes but still demanded time from BA, Development, and QA team members to assist the Application Support team.

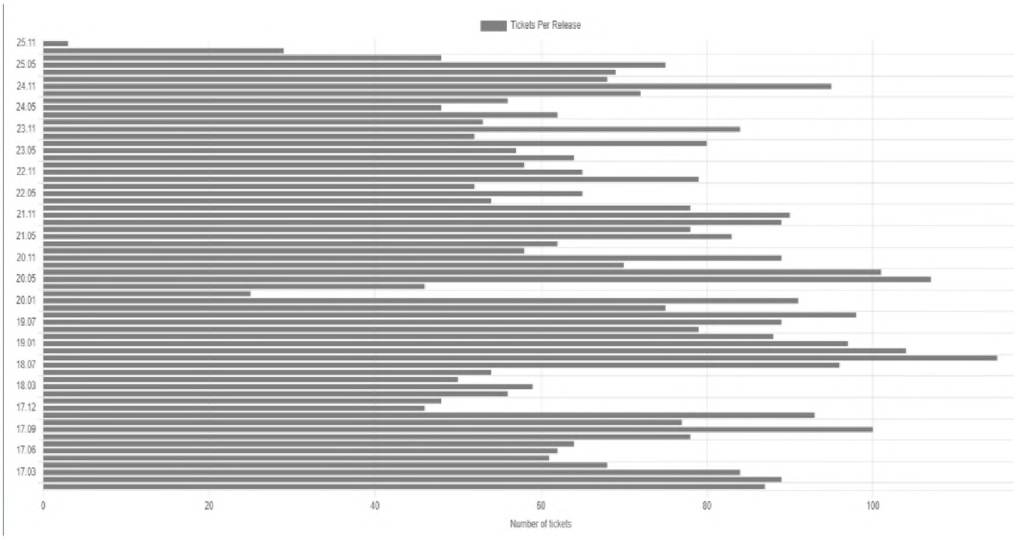

Figure 1. Number of Maintenance Tickets per 2-Month Release Cycle (2017–2024)

However, the dual-track model – Scrum for projects and Kanban for support – introduced new challenges. Release readiness often suffered, with stories being completed late in the cycle or converging simultaneously. Despite the two-month release cadence, planning and delivery were frequently misaligned. This misalignment also impacted release quality, at times necessitating emergency patches.

Figure 2 depicts the number of emergency patches issued annually from 2017 to 2024, showing a noticeable decrease following the adoption of Scrumban.

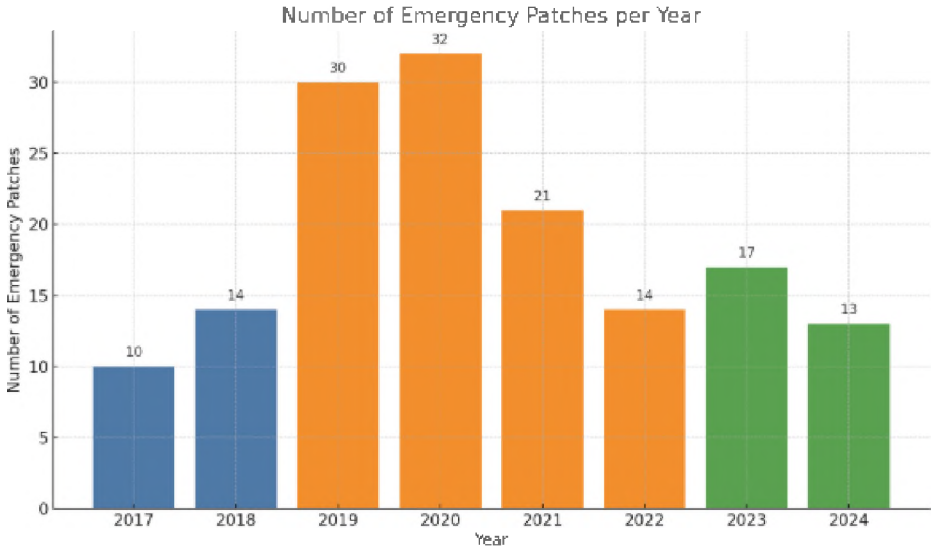

Figure 2. Emergency Patches Issued per Year (2017–2024)

The bars are color-coded by period:

- Blue for 2017–2018 (early years, pre-Agile shift)
- Orange for 2020–2022 (Scrum, Scrum + Kanban experimentation and transition)
- Green for 2023–2024 (post-Scrumban stabilization)

The long-term solution was a more consistent Scrumban approach applied across all development teams. Scrumban allowed Smart to retain the structured planning and role accountability of Scrum while benefiting from the flow-based task management of Kanban. This was particularly effective for Smart's mixed workload environment, where planned project features, regulatory deliverables, and reactive support work needed to be managed simultaneously.

Key elements of the adapted Scrumban framework included [5]:

- Flexible Planning Intervals: Instead of fixed sprints, teams conducted planning sessions every 2–3 weeks based on backlog readiness and release windows. This reduced pressure to force incomplete or ambiguous stories into a sprint cycle.
- Work-in-Progress (WIP) Limits: WIP thresholds were introduced per team and task type (e.g., "3 tasks max per developer" in dev stage), reducing multitasking and encouraging focus.
- Agile Checklists and "Definition of Done": All user stories had to meet minimum readiness criteria before being accepted into active work, including defined acceptance criteria, test scope, and documentation requirements.
- Pull-Based Task Assignment: Rather than assigning stories in sprint planning, team members selected work based on availability and skills, improving engagement and ownership.
- Adaptable Ceremonies: Teams retained retrospectives and demos but adapted their frequency – e.g., monthly retros for maintenance-heavy teams, and ad hoc reviews for large feature releases.

One of the defining characteristics of Smart's Scrumban implementation was the integration with semi-waterfall business analysis practices. BAs continued to conduct upfront requirement gathering and documentation for larger roadmap items. In most cases, development and QA work commenced only after 70-80% of BA analysis was complete. This helped ensure alignment with compliance standards and provided clarity for downstream teams. However, it also limited iteration opportunities during early implementation phases, and sometimes led to rework when assumptions changed post-handoff.

To bridge this gap, Smart introduced cross-functional planning reviews, where BAs, developers, and QAs discussed stories before they entered the WIP column. These reviews helped identify technical risks, testing implications, and scope conflicts earlier – bringing some iterative agility into the analysis pipeline.

Ultimately, Smart's shift to Scrumban delivered a balanced model: structured enough for long-term planning and compliance, but flexible enough to accommodate reactive work and shifting priorities. Though the system wasn't perfect – particularly during release crunch periods – it significantly improved transparency, reduced delivery stress, and fostered a culture of continuous improvement within the development organization.

5. Reflection: What Worked, What Didn't, and Implications for AI

5.1 What Worked in Agile Implementation

Smart's journey toward Agile maturity was marked by careful observation, adaptive planning, and practical decision-making. Among the most effective components were:

- Scrumban Framework: Adopting Scrumban allowed teams to maintain structured planning and reporting while enabling flow-based task handling. This was essential in Smart's context, where development and support work are continuously interleaved.
- Role-appropriate Tool Configuration: Once Jira was reintroduced and configured by someone with both technical and Agile expertise, it became a central tool for visibility, accountability, and reporting.
- Dedicated Teams for Key Projects: Projects where teams could remain focused—without being pulled into support or unrelated work—showed consistently higher sprint completion rates and better morale.
- Use of Kanban for Maintenance: Replacing Scrum with Kanban for maintenance helped reduce scope volatility, meeting fatigue, and average ticket resolution time.
- Cross-functional Planning: Including QA and developers in early backlog discussions helped reduce ambiguity and rework, especially for more complex features.
- Upfront BA Work (Waterfall-Like): While not traditionally Agile, this practice helped ensure compliance alignment and clarity. In most cases, development and QA work commenced only after 70–80% of BA analysis was complete. This reduced ambiguity and ensured traceability, though it sometimes reduced flexibility.

5.2 What Didn't Work Well

Despite the eventual success, several elements of Smart's Agile adoption journey presented persistent challenges:

- Initial Resistance and Misconfiguration: The first rollout of Jira failed due to lack of leadership support and the appointment of a Jira admin with little Agile experience. This set back adoption by several years.
- Overloaded Scrum for Maintenance: Attempting to manage all maintenance tickets using Scrum created bloated meetings, poor sprint adherence, and frustration, especially when team members had limited investment in non-project work.

• Split Resource Allocation: Teams fragmented across projects and roles often underperformed. Sprints were disrupted, priorities shifted mid-cycle, and backlog grooming suffered.

5.3 Recommendations for AI Implementation at Smart

Smart's management team is currently in process of developing internal policies and governing rules for the use of AI across the organization. This initiative includes defining ethical guidelines, model usage boundaries, data privacy standards, and roles responsible for oversight. The policy development is based on a combination of industry regulations and internal lessons from previous technology rollouts [2, 4]. Once finalized, these policies will provide the organizational foundation necessary for compliant and responsible AI experimentation—ensuring that innovation can move forward in a safe, auditable, and transparent way.

At the time of writing, Smart Solution has not yet deployed AI solutions in production, but several areas are under internal review and conceptual design. Accordingly, the following elements are presented as recommendations and strategic proposals – not as description of currently implement AI practices.

BA work at Smart typically occurs upstream of development and remains partially decoupled from the core Scrumban cycles, the BA team is well-positioned to drive early-stage AI planning. The following recommended actions would help align BA functions with future AI implementation:

• AI Readiness Assessment: Evaluate proposed AI use cases for business process fit, data availability, and anticipated user impact.

• Structured Problem Framing: Translate AI opportunities into clear business objectives and measurable outcomes, using hypothesis-driven templates.

• Data Governance Collaboration: Work closely with data engineers and legal/compliance teams to define acceptable data sources, consent rules, and retention policies.

• Iterative Specification Model: Replace full upfront requirements with milestone-based documentation tied to phases of AI model development (e.g., alpha, beta, production).

• Ethics and Explainability: Define risk scenarios and explainability requirements, along with fallback protocols in the event of model failure.

These steps would enable BAs to expand their scope beyond traditional documentation and contribute strategically across both Agile and AI discovery lifecycles.

As Smart begins exploring AI features—such as intelligent workflows, predictive analytics, or AI-assisted support—the organization can leverage its Agile experience to support future adoption. The following techniques are proposed based on lessons learned during Agile transformation.

Agile Discovery Practices

AI initiatives thrive in environments where experimentation and iteration are encouraged. Smart can use Agile techniques like:

• Lightweight story mapping and prototyping
• Iterative hypothesis validation (e.g., test an AI scoring before full-scale rollout)
• Short research & build cycles (2–3 weeks) to validate AI use cases

Dedicated Pilot Teams

Just as Smart saw success with focused Scrum teams, early AI projects should be assigned to cross-functional, **dedicated pods** (including data engineers, BAs, compliance experts, and developers) – mirroring the successful model used for early Agile projects.

Use of Agile Metrics

Monitor model iteration frequency, feedback from pilot users, and error rates to provide measurable ROI and transparency.

Compliance-First BA Framework

BA involvement in AI must grow beyond functional specs to cover ethical, legal, and data governance dimensions. Leveraging Smart's BA-first model, the company can create early documentation of:

- Data usage justification
- Risk scenarios
- Audit trails for AI decisions.

Avoid Tool Misfit

Just as Agile suffered from poor initial tooling, AI adoption should not rely on plug-and-play tools without context. Ensure internal champions are trained and external tools (e.g., model builders, analytics layers) are thoroughly vetted for Smart's technical and regulatory landscape.

6. Conclusion

Smart's Agile transformation - from early Scrum trials to a matured Scrumban model – demonstrates that successful adaptation is possible even in compliance-heavy, resource-constrained environments like banking software. The organization's journey reflects the importance of contextualizing Agile practices, aligning tools and roles with business realities, and prioritizing iterative learning over rigid adherence to frameworks.

As Smart begins its foray into AI-powered solutions, many of the principles that supported Agile success – cross-functional planning, incremental delivery, early validation, and clear governance – will serve as a foundation. However, AI adoption will require deeper collaboration between business analysts, data specialists, and compliance stakeholders.

This case study reinforces the idea that agility is not about tools or ceremonies alone, but about the organization's ability to align people, processes, and technology toward learning and adaptation.

References:

1. Anderson, D.J. (2010). Kanban: Successful evolutionary change for your technology business. Blue Hole Press.
2. Google. (2022). AI principles. https://ai.google/principles/
3. Highsmith, J. (2009). Agile project management: Creating innovative products. Addison-Wesley.
4. International Organization for Standardization. (2022). ISO/IEC 22989:2022 – Artificial intelligence – Concepts and terminology. ISO.
5. Ladas, C. (2009). Scrumban: Essays on Kanban systems for lean software development. Modus Cooperandi Press.
6. Rigby, D.K., Sutherland, J., & Takeuchi, H. (2016). Embracing agile. Harvard Business Review, 94(5), 40–50.
7. Schwaber, K., & Beedle, M. (2002). Agile software development with Scrum. Prentice Hall.
8. Stanford Institute for Human-Centered Artificial Intelligence. (2024). AI Index Report 2024. https://aiindex.stanford.edu/report/
9. Smart. (2017–2024). Internal documentation and retrospective reports (Unpublished internal documents).

Information about the author:

Evgenia Kulikova – corresponding author, MS Computer Science, Solutions Architect and Business Analysis Manager at Smart Solution in Ontario, Ontario, Canada; e-mail: ekulikova@smartsolution.com.