eoss-conf.com

ISSUE
№97

EUROPEAN OPEN
SCIENCE SPACE

COLLECTION OF SCIENTIFIC PAPERS

5TH INTERNATIONAL
SCIENTIFIC
AND PRACTICAL
CONFERENCE

SCIENTIFIC RESEARCH: MODERN INNOVATIONS AND FUTURE PERSPECTIVES

JULY 20-22, 2026, MONTREAL, CANADA

6. Felix M. AI-Powered Recommendation Systems as a Strategy for Reducing Decision Friction in Online Purchasing Journeys. 2026.
7. Rahman S. Enhancing Session-based E-commerce Recommendations with Self-Attention: A Transformer-based Approach. Modern Innovations, Systems and Technologies. 2026, 6(1), 3043–3050. https://doi.org/10.47813/2782-2818-2026-6-1-3043-3050
8. Lashchevska N., Cherevyk O. Analysis of Biomedical 3D Data Segmentation Methods Using Deep Learning. Наукові записки ДУІКТ. 2025, №1(7). https://doi.org/10.31673/2786-8362.2025.017489
9. Zhou G. et al. Deep Interest Network for Click-Through Rate Prediction. Proceedings of KDD 2018. ACM. https://dl.acm.org/doi/10.1145/3219819.3219823
10. Vaswani A. et al. Attention Is All You Need. Advances in Neural Information Processing Systems. 2017. https://arxiv.org/abs/1706.03762
11. Kang W.-C., McAuley J. Self-Attentive Sequential Recommendation. 2018 IEEE ICDM. https://doi.org/10.1109/ICDM.2018.00035
12. Covington P., Adams J., Sargin E. Deep Neural Networks for YouTube Recommendations. Proc. ACM RecSys 2016. https://doi.org/10.1145/2959100.2959190
13. Sun F. et al. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. Proc. CIKM 2019. https://doi.org/10.1145/3357384.3357895

IMPLEMENTING ARTIFICIAL INTELLIGENCE FOR SCRUM TEAMS MANAGEMENT

Znakhur Serhii
PhD, Associate Professor
Department of Information Systems
Znakhur Liudmyla
Senior Lecturer
Simon Kuznets Kharkiv National University of Economics,
Ukraine

Abstract

Agile methodologies, particularly Scrum, serve as the standard operating framework for over 95% of software organizations. However, mobile application development remains constrained by human cognitive biases, unpredictable estimation heuristic errors, device fragmentation challenges, and complex app store release lifecycles. This paper presents a novel Hybrid AI-Agile Framework specifically engineered to optimize Scrum team management within mobile systems development. The proposed framework integrates predictive machine learning models, natural language processing (NLP), and multi-agent Large Language Model (LLM)

architectures to automate and augment critical project management processes, including backlog refinement, sprint planning, risk assessment, dynamic resource allocation, and retrospective feedback analysis. We detail the operational mechanics of our framework and validate its efficacy through a controlled, 50-task experimental benchmark simulating a full mobile systems development backlog. The study establishes a reproducible paradigm for integrating cognitive automation with human-in-the-loop oversight to reduce IT project failure rates in mobile systems engineering.

Keywords: Scrum Management, Artificial Intelligence, Multi-Agent Systems, Agile Software Development, Large Language Models (LLMs), Predictive Analytics.

Introduction

The global software landscape has undergone a tectonic shift toward mobile-first architectures. Mobile systems are characterized by intense hardware fragmentation, diverse operating system lifecycles, strict memory and power constraints, and a highly volatile user-feedback ecosystem governed by the App Store and Google Play marketplaces. Managing software engineering teams in this environment demands exceptional adaptability, rapid release cycles, and rigorous quality control. Traditional software development lifecycles (SDLC) have largely been replaced by Agile methodologies, with Scrum being the dominant framework. Indeed, by 2025, over 95–97% of modern IT organizations have incorporated Agile practices to some degree. Yet, despite near-universal adoption, traditional project management frameworks in mobile software engineering continue to experience high failure rates. Recent industry analyses indicate that only 16–30% of IT and software projects fully succeed on time, within budget, and within scope, with mobile application projects showing particularly high rates of sprint failures, regression defects, and post-release crashes due to human error in estimation, prioritization, and dependency mapping. Traditional Scrum team management relies heavily on human heuristics, subjective expert estimates, and manual tracking of backlog items. Project managers, Scrum Masters, and Product Owners must synthesize massive quantities of unstructured data—ranging from user feedback, app analytics, crash logs, and pull requests to team velocity charts and developer communications—while navigating cognitive biases, such as the planning fallacy and anchoring bias. In the mobile domain, these challenges are compounded by the need to manage cross-functional teams, including backend developers, iOS specialists, Android developers, and UI/UX designers, whose tasks are tightly coupled yet technically distinct. The rapid maturation of Generative Artificial Intelligence (AI), agentic multi-agent systems, and advanced Large Language Models (LLMs) offers an unprecedented opportunity to address these systemic limitations. State-of-the-art architectures, including Claude 4, Gemini 2.5 Pro, DeepSeek, and specialized coding models, can reliably emulate crucial roles within the Scrum ecosystem. These models transcend basic code autocomplete; they act as autonomous, collaborative agents that can serve as Product Owners, Technical Architects, Automated Code Reviewers, and Risk Analysts. By analyzing vast project histories and executing predictive algorithms, AI-Augmented Project Management can deliver high accuracy in risk prediction, cycle-time compression, and effort estimation. Despite these technological

advancements, the integration of AI into Agile and Scrum processes remains fragmented. Most organizations restrict AI usage to isolated plugins (e.g., GitHub Copilot or basic Jira AI summarization) rather than adopting a cohesive, disciplined, and multi-agent project management architecture. This creates a significant "utilization gap," with only 12–25% of organizations leveraging AI substantially in their project management routines. This study addresses this gap by presenting a modular, scalable, and mathematically grounded Hybrid AI-Agile Framework specifically adapted for mobile systems development. By implementing a multi-agent LLM ecosystem that respects the core values of the Agile Manifesto—empiricism, transparency, inspection, and adaptation—we demonstrate how organizations can unlock superhuman execution speeds and consistency without sacrificing human collaboration and ethical oversight [1, 10].

Aim and objectives of the research

The primary objective of this research is to design, implement, and empirically validate a modular, multi-agent AI-driven decision support system (DSS) designed to optimize Scrum team management and delivery in mobile systems engineering, ensuring high predictability, resource utilization, and product quality under conditions of uncertainty.

To achieve this aim, the following tasks were defined and completed:

1. Perform a comprehensive review of existing AI applications in project management, isolating the unique constraints and requirements of mobile system development lifecycles.
2. Architect a modular, Agile-compatible multi-agent framework featuring specialized virtual roles (Product Owner Proxy, Mobile Developer Agent, Technical Code Reviewer, and Scrum Master Facilitator) operating in a cohesive workflow.
3. Develop and validate mathematical formulations for real-time risk scoring and dynamic resource allocation tailored to the high-volatility mobile development environment.
4. Implement and test a suite of 50 representative development and machine learning tasks within the framework, executing multi-iteration simulations to assess performance.
5. Execute a comparative study pitting the AI-augmented multi-agent system against high-performing human development teams across identical tasks, utilizing objective software metrics (throughput, consistency, defect rate, and compliance).
6. Formulate a phased adoption playbook and maturity model to guide organizations in transition from traditional Scrum to AI-hybrid project management paradigms.

Literature review

The intersection of software development methodologies, user-centered design (UCD), and artificial intelligence represents a rich area of academic and industry research. To ground our hybrid framework, we analyze the historical trajectory of these approaches, highlighting their evolution and identifying critical gaps in their application to modern mobile systems.

Agile methodologies arose as a direct rejection of waterfall models, prioritizing iterative releases and user feedback. While exceptionally successful in standard enterprise software, Agile faced significant integration friction when combined with User-Centered Design (UCD) and User Experience (UX) processes—concepts that are vital for mobile systems where user interfaces are the primary determinant of application adoption. Systematic reviews of Agile and UCD integration (AUCDI) show that these challenges span across infrastructure, personnel, and workflows. Key bottlenecks include a lack of time for upfront research (leading to rushed, fragmented user interfaces), difficulty modularizing large-scale design tasks into sprint-sized "chunks," and the logistical complexity of scheduling usability testing within compressed 2-week iterations. These challenges are exacerbated in information systems where a single interface change must scale across hundreds of different screen sizes, aspect ratios, and operating system variations [1, 3-9].

Table 1. Agile Adoption Statistics across Industries (Historical Trend to 2025)

Methodology / Framework	Adoption Rate (2023)	Adoption Rate (2025)	Primary Mobile System Application Focus
Scrum	56%	65%	Cross-platform sprint coordination (iOS/Android)
Kanban	5%	8%	Continuous deployment pipelines, hotfixes, and patch management
ScrumBan / Hybrid	8%	12%	Complex mobile client-server application architectures
Scrum/XP Hybrid	6%	9%	Mobile SDK development, strict test-driven development (TDD)
Interactive/Iterative	3%	2%	Early-stage prototyping and UI layout verification
Other Frameworks	22%	4%	Legacy systems integration and non-standard mobile architectures

The application of Artificial Intelligence within project management has historically evolved from rule-based expert systems in the 1980s to basic machine learning models for risk analysis in the 2010s. The true paradigm shift occurred with the maturation of transformer-based LLMs and agentic AI architectures between 2023 and 2025. Today, AI-driven project management is no longer merely a predictive tool for project scheduling; it serves as an active participant in project execution. Industry reports indicate that the AI in project management market is growing at a CAGR of 16.3%, projected to expand from USD 3.08 billion in 2024 to USD 7.4 billion by 2029. In the tech sector, approximately 78% of organizations use AI for at least one function, and early adopters have demonstrated up to 25% reductions in project delays and 15–20% increases in sprint velocity through predictive modeling. AI-enabled platforms

can analyze historical delivery logs and automatically forecast risk profiles with up to 94% accuracy, significantly outperforming subjective human planning.

However, current approaches fail to address the specific needs of mobile system development. Traditional AI integrations inside project management tools (predictive sprint charts in Jira) do not account for the complexities of mobile-specific environments, such as cross-platform code parity, target SDK updates, API deprecation, app store compliance guidelines, and on-device machine learning integration (PyTorch Mobile, ONNX). Furthermore, standard AI models do not support the highly collaborative multi-agent setups required to mimic the complex interactions of a Scrum team. Our framework addresses these exact gaps by tailoring multi-agent roles and workflow priorities directly to mobile systems engineering, leveraging local, high-performance LLM deployment architectures (Ollama, Mixtral, CodeLlama) to ensure strict data privacy and offline operational capabilities.

Proposed Hybrid AI-Agile Framework

The proposed Hybrid AI-Agile Framework is a multi-agent decision support system designed to automate and optimize the lifecycle of mobile systems development. It consists of three primary, tightly coupled layers: the Data Collection and Integration Layer, the Multi-Agent Processing Layer, and the Process Execution and Feedback Layer. By leveraging a local Ollama server running specialized LLM configurations, the framework ensures absolute control over enterprise source code and project metrics, establishing a secure environment compliant with GDPR and strict internal data policies.

The architectural framework is organized around specialized agents that collaborate sequentially to mimic a physical Scrum micro-team. These roles are defined as follows:

1. **Mobile Product Owner (PO) Agent:** This agent ingests high-level, unstructured customer requirements, OS specifications, and store feedback to auto-generate structured, INVEST-compliant user stories. It maps out dependencies between mobile clients (iOS/Android) and backend API endpoints, and defines rigorous acceptance criteria.

2. **Mobile Developer Agent.** Tailored with specialized system instructions for mobile systems (e.g., Swift for iOS, Kotlin for Android, and cross-platform frameworks like Flutter/React Native). It produces clean, modular, and optimized code, utilizing proper error-trapping patterns, modern networking clients, and thread-safe operations.

3. **Technical Code Reviewer Agent** This agent conducts thorough evaluations of the developer's output. It verifies compliance with SOLID design principles, mobile memory-optimization best practices, and security standards (such as secure keychain storage and certificate pinning), and evaluates structural alignment with the original acceptance criteria.

4. **Scrum Master Facilitator Agent.** Coordinates task orchestration, monitors cycle times, and evaluates project risks. The Scrum Master Agent implements a mathematical risk mitigation and resource allocation model designed to prevent project overruns.

The operational risk score R_i for any given mobile development task i within a sprint is dynamically calculated as the product of its risk probability and impact:

$$R_i = P_i \times I_i \quad (1)$$

where P_i represents the probability of risk occurrence (e.g., code regressions, API incompatibilities, or app store rejections). This probability is modeled using a logistic sigmoid function based on continuous operational indicators:

$$P_i = 1/(1 + e^{-zi}) \quad (2)$$

Here, the parameter z_i represents a weighted linear combination of core quality indicators, platform dependencies, and test coverage metrics. The coefficient values are calibrated from historical project repositories:

$$z_i = \beta_0 + \beta_1 \cdot C_i + \beta_2 \cdot D_i - \beta_3 \cdot T_i \quad (3)$$

where C_i represents code complexity (measured via cyclomatic metrics), D_i denotes platform-specific API dependency factor, and T_i represents automated unit and UI test coverage. The risk impact, I_i, is a normalized scale (0.0 to 1.0) representing the estimated sprint delay or time-overrun impact if the risk manifests.

To allocate developer resources dynamically, the framework runs an optimization algorithm. Let x_{ij} be a binary decision variable indicating whether task i is allocated to developer resource j. The Scrum Master agent seeks to minimize total sprint execution risk across N tasks and M developers:

$$\text{Minimize} \sum_{(i=1 \text{ to } N)} \sum_{(j=1 \text{ to } M)} R_i \cdot t_{ij} \cdot x_{ij} \quad (4)$$

subject to developer capacity constraints (no developer can be assigned more than their maximum sprint hours capacity, typically representing a maximum of five simultaneous active tasks to avoid context-switching overhead) and a minimum risk-weighted task priority threshold. By solving this linear programming model in real time, the system balances workloads and prevents resource overutilization, achieving an average workload balancing efficiency of 92–94% in simulated environments.

This structured approach ensures that any mobile development process transitions from a reactive, human-centered model into a highly predictable, data-driven system. The flow of tasks, code artifacts, and risk indicators is visualized in the project's orchestration loop, showing how agent outputs feed directly into automated pipelines with human-in-the-loop validation gates.

Results of the research

To validate the proposed framework, a comprehensive controlled experiment was executed. We established an Agile backlog consisting of 50 complex tasks representative of modern mobile systems and on-device machine learning engineering. The experiment was conducted using an Ollama local deployment. The Product Owner and Reviewer agents utilized the Mixtral-8x22B-Instruct model (128k context window) to support intensive reasoning and contextual mapping, while the Developer agent used CodeLlama-34B-Instruct. The physical setup comprised a single workstation equipped with an NVIDIA RTX 2060 GPU (16 GB VRAM). All runs were

executed with a controlled temperature setting of 0.2 to ensure reproducibility. For each task, the multi-agent system executed 5 complete iterations.

Table 2. The 50 Tasks (Experimental Backlog)

Task	Task Description (User Story Core)	Complexity	Expected Output
1	Bash script wrapper for Gradle build automation & Fastlane deployment	Low	Production-ready deployment automation script
2	Telemetry and crash log preprocessing pipeline (Pandas + Scikit-Learn)	Medium	Jupyter data processing script + analytics tool
3	Fine-tune DistilBERT for App Store user review sentiment analysis	High	HuggingFace fine-tuning model + python script
4	Implement offline on-device mobile RAG system using LlamaIndex	High	Embedded database query interface and local API
5	Build PyTorch Mobile computer vision pipeline for on-device CIFAR-10	High	Model exporter + mobile-optimized runtime pipeline
6	Create LangChain agent for autonomous on-device Web automation	Very High	Fully autonomous mobile browser automation script
7	Deploy high-throughput FastAPI + Streamlit backend for mobile apps	High	Dockerized API endpoint container architecture
8	Privacy-preserving on-device Federated Learning (Flower framework)	Very High	Multi-client client-server learning simulator
9	Build MLflow telemetry comparison dashboard for mobile performance	Med-High	Telemetry monitoring dashboard + model registry
10	Build reinforcement learning agent for dynamic mobile ad targeting	High	Trained agent + simulation deployment code
.	.	.	.
50	Build MLflow metrics of performance	Med	Dashboard of metrics

The performance of each task was evaluated across seven dimensions. User Story Quality was verified by human editors who checked the INVEST guidelines. Code Functionality was assessed through automated test-harnesses (pytest) and manual compilation. Code Quality & SOLID compliance was extracted via SonarQube analysis. Best Practices Compliance checked for explicit error handling, proper logging, and type hints. Execution Time was captured using built-in system timers. Consistency represents the standard deviation of score variances across 5 runs, and the Overall Sprint Score was computed as a weighted average: 40% code functionality, 30% code quality, 20% user story completeness, and 10% execution velocity.

Table 3. Experimental Results for the 50 Tasks (Averaged Over 5 Runs)

Task	Risk Score (R)	Prob (P)	Impact (I)	Story Score	Code Func	Code Qual	Best Pract	Time (s)	Consistency (σ)	Overall Score
1	0.36	0.90	0.40	9.4	10.0	9.6	9.8	68	0.14	9.72
2	0.27	0.90	0.30	9.2	9.8	9.4	9.5	92	0.22	9.55
3	0.28	0.70	0.40	9.6	10.0	9.7	9.9	214	0.18	9.81
4	0.24	0.80	0.30	9.2	9.6	9.3	9.5	285	0.31	9.41
5	0.40	0.80	0.50	9.3	9.9	9.5	9.8	178	0.19	9.64
6	0.36	0.90	0.40	8.9	9.4	9.1	9.3	342	0.42	9.21
7	0.36	0.60	0.60	9.3	10.0	9.8	9.9	156	0.16	9.83
8	0.29	0.95	0.30	8.8	9.2	8.9	9.1	398	0.51	9.05
9	0.40	1.00	0.40	9.4	9.9	9.6	9.8	134	0.21	9.70
10	0.30	0.65	0.7	9.0	9.0	9.5	9.2	167	0.18	9.58
.	.	.	.	.	.	.	.	.	.	.
50	0.20	0.50	0.40	9.2	9.7	9.4	9.6	189	0.27	9.53
AVG	0.32	0.80	0.40	9.24	9.75	9.43	9.62	205.6	0.26	9.55

The data compiled in Table 3 reveals an outstanding high-performance profile. The multi-agent AI system achieved an average Overall Sprint Score of 9.55 out of 10. This is exceptionally high and equates to the output quality of elite, senior-level human engineers. The average sprint cycle execution time was 205.6 seconds (approximately 3 minutes and 25 seconds). Traditional Scrum processes involving backlog prioritization, user story drafting, actual coding, and multi-stage manual reviews typically require 6 to 12 hours of human labor per story in real-world environments. The fastest task execution was achieved for Task 1 (Gradle & Fastlane bash wrapper) in 68 seconds. The longest and most computationally expensive task was Task 8 (privacy-preserving federated learning simulation), at 398 seconds, and it also exhibited the highest variance in consistency ($\sigma = 0.51$) due to structural and algorithmic complexity. High-performing tasks like Task 7 (FastAPI deployment) scored a near-perfect 9.83, reflecting the model's capacity for highly standardized backend development.

To establish comparative rigor, we conducted a parallel study pitting our AI framework against 5 senior-level human software engineering teams. These teams, each consisting of a dedicated Scrum Master, Product Owner, and developers, completed the same 50 tasks over a two-week period. The comparison was based on the same key metrics, and the results are detailed in Table 4.

Table 4. Metric Comparison between AI-Augmented Crew and Human Agile Teams

Evaluation Dimension	AI Multi-Agent Crew	Senior Human Teams	Performance Delta (%) / Improvement
Average Overall Performance Score	9.55 / 10	8.41 / 10	+13.6% quality improvement
Average Time per Task	205 seconds	9.4 hours	164x speed acceleration
Consistency (σ across runs)	0.26	0.94	3.6x more consistent output
First-time-correct Rate	92%	64%	+43.8% defect reduction

The data presented in Table 4 confirms several advantages of the multi-agent AI system. While senior human teams generated highly contextualized designs and showed deep contextual understanding of Edge OS parameters, they struggled with planning consistency and were limited by fatigue and estimation biases. The AI crew completed tasks in an average of 3.4 minutes, compared to the human average of 9.4 hours per task—representing a 164x speed acceleration. This massive throughput allows mobile project managers to execute pre-sprint simulations, verifying the technical feasibility of hundreds of feature ideas within an hour before committing human resources to development. Furthermore, the first-time-correct rate of 92% (compared to the human 64%) highlights the performance of integrated AI code reviewers in detecting syntax, memory leaks, and architectural deviations before compiling code on physical devices.

The mathematical risk mitigation model proved effective during the simulations. Under conditions of team capacity constraints, the resource optimization engine dynamically balanced task allocations. When testing with variable task complexities, the system adjusted allocation variables x_{ij}, maintaining a stable 92% workload distribution and protecting the simulated engineering pipeline from bottleneck blockages. This provides robust empirical proof that orchestrated LLM ensembles possess the maturity to automate significant portions of Agile sprint cycles in highly technical knowledge domains, such as mobile systems development.

Conclusions

This research investigated the systematic implementation of artificial intelligence within Scrum team management for mobile systems development. We developed and evaluated a Hybrid AI-Agile Framework designed to automate and augment traditional project management practices through predictive risk analysis, resource optimization algorithms, and multi-agent LLM systems.

The key conclusions derived from this study are summarized as follows:

1. Traditional Scrum processes are limited by human cognitive biases and high manual transaction costs, causing project delays and estimation inaccuracies. Integrating AI offers a scalable pathway to optimize decision predictability and execution speed.

2. The proposed Hybrid AI-Agile Framework utilizes specialized multi-agent roles

(Product Owner, Developer, Reviewer, Scrum Master) orchestrated via a local Ollama server. This architecture maintains strict enterprise data security while establishing real-time feedback loops across the mobile application development lifecycle.

3. Controlled experimental validation on a backlog of 50 complex development and machine learning tasks demonstrated the system's efficiency. The AI crew completed sprint tasks with an average overall performance score of 9.55/10 and achieved a first-time-correct rate of 92%.

4. Comparative analysis with senior human development teams showed that the AI-augmented system achieved a 164x speed acceleration (completing tasks in 3.4 minutes versus 9.4 hours) and a 3.6x improvement in output consistency, showing its potential as an execution environment for rapid prototyping and simulation.

5. The mathematical formulation for dynamic risk calculation ($R_i = P_i \times I_i$) and linear resource optimization proved effective in balancing simulated team capacity, maintaining high resource utilization (92–94%) and mitigating execution risks.

However, important limitations must be acknowledged. First, the experimental runs were executed in controlled, simulated environments with synthetic backlogs, potentially underrepresenting the social, relational, and cross-functional communication complexities present in physical organizations. Second, the cold-start phase of LLMs—including initial context configuration—can introduce latencies in real-world deployment, and potential bias in the underlying training weights requires continuous manual audit. Finally, scaling the multi-agent system to large, enterprise-wide portfolios (e.g., SAFe) may introduce significant computational and token-hosting costs that must be balanced against the project's financial return.

Future research should focus on optimizing on-device project management LLMs to run directly on secure mobile environments, integrating real-time user-market telemetry data into the Product Owner agent's prioritization loops, and conducting longitudinal industrial field studies to track the long-term impacts of AI-augmented project management on human team morale and organizational productivity.

References

1. Ambler, S. (2008). Tailoring Usability into Agile Software Development Projects. In E. Law, E. Hvannberg, & G. Cockton (Eds.), Maturing Usability (Human-Computer Interaction Series, pp. 75–95). Springer. https://doi.org/10.1007/978-1-84628-941-5_4
2. Beck, K. (2000). Manifesto for Agile Software Development. http://agilemanifesto.org
3. Blomkvist, S. (2005). Towards a Model for Bridging Agile Development and User-Centered Design. In A. Seffah, J. Gulliksen, & M. Desmarais (Eds.), Human-Centered Software Engineering - Integrating Usability in the Software Development Lifecycle (Human-Computer Interaction Series, Vol. 8, pp. 219–244). Springer. https://doi.org/10.1007/1-4020-4113-6_12
4. Chamberlain, S., Sharp, H., & Maiden, N. (2006). Towards a Framework for Integrating Agile Development and User-Centred Design. In Proceedings of the 7th

International Conference on Extreme Programming and Agile Processes in Software Engineering (XP'06) (pp. 143–153). Springer. https://doi.org/10.1007/11774129_15
5. Cruzes, D., & Dyba, T. (2011). Recommended Steps for Thematic Synthesis in Software Engineering. In Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement (ESEM '11) (pp. 275–284). IEEE Computer Society. https://doi.org/10.1109/ESEM.2011.36
6. da Silva, T. S., Martin, A., Maurer, F., & Silveira, M. (2011). User-Centered Design and Agile Methods: A Systematic Review. In Proceedings of the Agile Conference (AGILE'11) (pp. 77–86). IEEE. https://doi.org/10.1109/AGILE.2011.24
7. Duchting, M., Zimmermann, D., & Nebe, K. (2007). Incorporating User Centered Requirement Engineering into Agile Software Development. In Proceedings of the 12th International Conference on Human-Computer Interaction: Interaction Design and Usability (HCI'07) (pp. 58–67). Springer. https://doi.org/10.1007/978-3-540-73105-4_7
8. Fox, D., Sillito, J., & Maurer, F. (2008). Agile Methods and User-Centered Design: How These Two Methodologies are Being Successfully Integrated in Industry. In Proceedings of the Agile 2008 Conference (AGILE '08) (pp. 63–72). IEEE Computer Society. https://doi.org/10.1109/Agile.2008.78
9. Cohn M. User Stories Applied: For Agile Software Development / M. Cohn. – Boston : Addison-Wesley, 2004. – 268 p.
10. Rubin K. S. Essential Scrum: A Practical Guide to the Most Popular Agile Process / K. S. Rubin. – Boston : Addison-Wesley, 2012. – 452 p.

DOI 10.70286/EOSS-20.07.2026.009.151-156

AN LLM-BASED DECISION INTEGRATOR WITH STRUCTURED OUTPUT AND A SIX-GATE ANALYTICAL RUBRIC FOR A/B TESTING DECISION SUPPORT

Moskovskykh Andrii
Master of Computer Sciences

Abstract. The statistical layer of a modern experimentation pipeline produces more numbers than decisions, and integrating its dozen signals into one defensible call has remained manual work. The paper proposes a decision integrator built on a large language model constrained twice: structured output against a fixed JSON schema closes the space of recommendations to four labels (ship, do not ship, keep running, investigate), and a six-gate analytical rubric in an immutable versioned system prompt fixes the order in which evidence is weighed, so no later gate can override an earlier one. A deterministic template guarantees a baseline report when the model is unavailable. Over two months of production use all 20 calls returned schema-valid