2025 13th International Conference on Intelligent Embedded, MicroElectronics, Communication and Optical Networks (IEMECON) | 979-8-3315-7201-3/25/$31.00 ©2025 IEEE | DOI: 10.1109/IEMECON69302.2025.11365878

Smart Agile Planning: A Knowledge-Grounded Multi-Agent Framework

Archita Dasgupta
Department of Computer Science and Engineering
Institute of Engineering and Management
Kolkata, India
archita.dasgupta2022@iem.edu.in

Saikat Dutt
Department of Computer Science and Engineering
Institute of Engineering and Management
Kolkata, India
saikat.dutt@iem.edu.in

***Abstract*—Agile practices tend to suffer bottlenecks due to the reason that they are manually carried out and predominantly due to the ad-hoc nature of sprint and release planning. This paper explains an AI-based framework that utilizes a Retrieval-Augmented Generation (RAG) system to implement, automate and enable end-to-end planning intelligently. Our system is based on a multi-agent framework with every agent handling a specific activity: backlog estimation, release planning and sprint planning. With past organizational knowledge and sequential agent workflow, our system addresses main issues of efficiency, consistency and scalability in agile environments. Our research indicates that AI not only automates repetitive administrative work, it also assists in systematic backlog estimation and sprint planning from real data ensuring consistency, equity, and transparency in decision-making for software development teams sticking to ethical principles. Our framework minimizes manual planning time, enhances the accuracy of estimation, and yields more consistent planning results than traditional manual methods.**

***Index Terms*—Sprint Automation, Lang-graph, RAG, Agentic AI, Multi-agent framework**

I. Introduction

Agile methodologies have been one of the most used approach in software development. Agile's iterative and adaptive practices, for example, Scrum and Kanban [1], were all the time found as most successful in dealing with the changing environment, where the team can continuously provide value. The essence of Agile is about project planning which is modified after each sprint cycle and involves three interdependent activities: backlog estimation, release planning, and sprint planning. Agile, however, does not come without limitations. For example, the short duration of a sprint cycle, usually, is 2-4 weeks and has very little time for in-depth planning. This situation leads to rushed estimations being performed, the incomplete breakdown of the tasks, commitments being unrealistic, and dependencies going unnoticed. If people leave the team or the team changes then the process is weakened again. The senior members' departure leads to the loss of valuable historical knowledge, while newer members find it hard to engage productively in planning, thus, lowering the accuracy and reliability. Moreover, Agile planning is still a very manual process and it is dependent on the use of spreadsheets and repetitive administrative updates which are error-prone, inefficient and lack transparency. We have a vision of an AI-driven framework that can revolutionize Agile planning in order to handle these drawbacks. The features of a multi-agent architecture give us the opportunity to allocate the role of each agent backlog estimation, release planning, or sprint scheduling. Retrieval-Augmented Generation (RAG) is the method used to bring the organizational knowledge bases closer, including historical backlogs and the profiles of team skills. By coordinating agents as a seamless workflow from the highest level of estimation to the most detailed task, the framework improves the effectiveness, the consistency, and the scalability. Our work casts AI not just as a tool for automation but as a facilitator of more methodical, data-driven, and equitable Agile planning, which ultimately makes it possible for the development teams to concentrate on creating the value.

II. Background Study

The convergence of Agile methods and AI represents a disruptive paradigm shift in software development that is characteristic not only of data-driven but also of reactive to proactive, augmented, and data-driven methods. The articles discussed provide a firm theoretical foundation on which to understand this transformation. One of the main ideas is Agentic AI, which is described as having the characteristics of independence, initiative, and the ability to learn, as per the definition of Hosseini and Seilani (2025) [3], who make a point of its function to be that of a support for the automatic implementation of processes that lead to the increase of the efficiency of the organization. Apró (2025) [4] brings the Agentic AI for Proactive IT Forecasting (AAPIF) model as the main feature that draws on reinforcement learning and multi-source data fusion to anticipate project results, thus demonstrating an important contrast with conventional methods that rely on "siloed, retrospective data" and are usually reactive. The idea has also been elaborated in the context of multi-agent systems, where several autonomous AI agents with different specializations dynamically interact. Khan and Daviglus (2025) [5], Abbas and Wahab (2024) [6], and Nasir and Hussain (2024) [11] explore how these systems streamline workflows and enhance team collaboration through tasks like code generation, bug detection, documentation, and sprint planning. They consistently find that these systems reduce development time, enhance productivity, and provide real-

979-8-3315-7201-3/25/$31.00 ©2025 IEEE

time decision support, with Abbas and Wahab (2024) [6] specifically noting the benefits of "automated backlog prioritization" and "real-time code reviews". This collaboration model is seen as an augmentation of human roles rather than a replacement. Parikh (2025) [7] proposes a "co-evolutionary framework" for product management where the PM's role shifts to an "orchestrator," requiring new skills in AI literacy, governance, and systems thinking. Khan and Kallinteris (2025) [10] reinforce this by presenting a framework where multi-agent LLMs are designed to "augment, rather than replace, human developers," fostering a symbiotic relationship. Furthermore, Jordan (2025) [9] demonstrates how generative AI-enhanced conversational agents can automate critical tasks across the project lifecycle, such as generating work breakdown structures and producing real-time status reports, which improves accuracy and decision-making. Sawant (P.D.) [8] presents a high-level vision of a unified, self-learning "Agentic AI ecosystem" that integrates diverse data sources and automation tools for continuous, real-time analysis, demonstrating its transformative potential for maximizing operational efficiency. Despite these advantages, the literature provides a balanced view by critically examining the hurdles to successful adoption. Sanches (2025) [2] points to "limited compatibility with legacy systems," data privacy, and "cultural resistance to change" as major obstacles. Other papers, including Khan and Daviglus (2025) [5], Khan and Kallinteris (2025) [10], and Nasir and Hussain (2024) [11], raise concerns about "model interpretability," "error propagation," and "hallucinations," which are AI-generated falsehoods that threaten the reliability of AI-generated code. These challenges, including the need to address "ethical considerations" and integrate with "existing DevOps pipelines," underscore the need for further research, particularly through industry-specific case studies.

III. Methodology

A. Problem Statement

Today, more than 80% of software projects adopt Agile methodologies. Agile relies on iterative planning. So, the planning activities are revisited in every sprint. The key Agile planning activities include Product Backlog estimation, Release Planning, and Sprint Planning. The typical steps are:

- Estimating the Product Backlog in Story Points
- Distributing the product backlog into multiple releases based on the release timelines and feature priorities, forming a release backlog
- Breaking the release backlog into multiple sprints according to team capacity and composition, creating a sprint backlog
- Defining tasks for each user story and assigning them to team members based on competency. This granular plan becomes the basis for ongoing progress tracking

However, the teams face several recurring challenges when performing these planning activities with the same rigor each sprint:

1) Time constraints: Short sprint cycles often leave limited room for in-depth planning

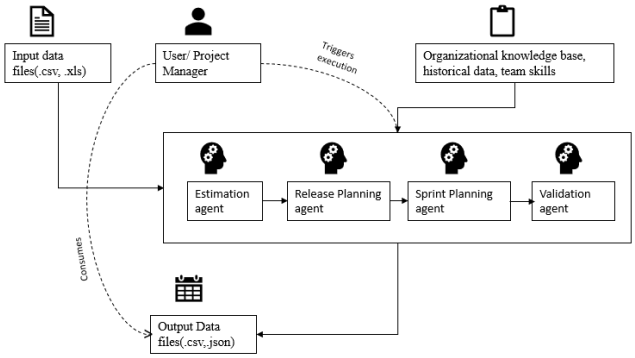

Fig. 1. System Architecture

2) Team churn: Frequent changes in team members can erode historical estimation and benchmark knowledge
3) Inexperience: Junior team members may struggle to participate in estimation and planning activities
4) Manual effort: Repetitive manual tasks increase the risk of errors

B. System Architecture

In Agile software development, the sprint is the basis for iterative progress, typically two to four weeks in duration in which a cross-functional team commits to the completion of an identifiable list of backlog items. Although the framework provides flexibility, the sprint planning is often a bottleneck, involving gigantic manual effort in estimating story points, allocating work, balancing constraints, and feasibility checking. This is often seen in large projects where organizational comprehension, team velocity, and dependencies must be simultaneously in dynamic equilibrium. Our solution to address this suggests a new AI-aided sprint automation platform that maximizes the entire planning cycle. As indicated in Fig. 1, the system begins by analyzing input data files (CSV, Excel) along with the project backlog, past performance data, team organization, and release constraints. This information is cross-checked against the organizational knowledge base, which keeps track of historical backlog data, previous estimated story points, completion dates, and team skill matrices. Together, they provide the context needed for realistic and accurate sprint planning. The workflow is started by the user or project manager, initiating execution through the interface. When called, the multi-agent pipeline process is what is passed on and serves as the system's central intelligence. The pipeline involves four dedicated agents operating in sequence: the Estimation Agent produces story point estimates according to previous analogies; the Release Planning Agent converts these estimates into a sprint-by-sprint allocation of the backlog items, considering velocity and constraints; the Sprint Planning Agent slices these items into low-level tasks and allocates them to roles according to skills and capacity; and finally, the Validation Agent checks the feasibility and safety of the plan, flagging unrealistic or infeasible output. The output of

this process is formalized as structured artifacts such as CSV and JSON files. These are the projected backlog with rationale, release-level sprint allocations, and full sprint plans with task owners and effort hours. These artifacts not only reduce the workload on hand planning but also make it transparent and verifiable, keeping the sprint planning reality-based and based on organizational reality.

1) Data Abstraction Layer: The Data Abstraction Layer (DAL) is the base of the proposed sprint automation system(fig 2. Its basic role is to import,clean, and normalize inputs from heterogeneous organizational repositories, hence enabling higher-level agents to operate on a single and reliable data model. While the DAL is concealing the differences in the source format, it still enables the other processes, especially the estimation and planning agents, to work with the same interface, to maintain their integrity, and to perform some auditing functions. The DAL ensures that rest of the

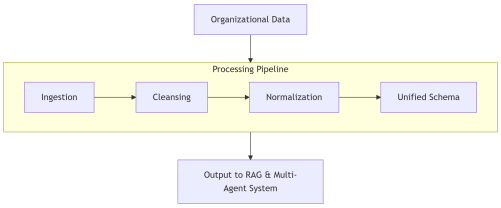

Fig. 2. Data Abstraction Layer

system—especially the RAG-driven Estimation Agent operates on clean, structured, and context-rich data. It is the system's data pipeline backbone, coverting organizational artifacts into machine-readable representation allowing proper retrieval, reasoning, and decision-making.

2) Retrieval-Augmented Generation (RAG) for Sprint Automation: The RAG implementation in our architecture (fig 3 is the main differentiator between a naïve "Chat-GPT"-like sprint planner and a solid, organization-based system. Contrary to common LLM answers, our system minimizes hallucination [14], maintains fact grounding, and follows historical organizational data to generate correct estimations and schedules.

a) Data Chunking Vectorization (Offline Preprocessing): **Text Splitting:** The project's historical backlog consisting of user stories, their related metadata (actual vs. estimated effort), and sprint results are divided into smaller chunks through *CharacterTextSplitter*(a text-splitter component that divides big text documents into smaller chunks according to character number.).

- We use [*chunk_overlap=100*] so that semantically related entities (for example a user story and acceptance criteria, or execution logs with an estimate) remain together.
- Such overlap is essential to maintain contextual integrity without giving a chance to a LLM to hallucinate relationships from incomplete chunks.

Embedding Model: Every chunk is converted to high-dimensional embeddings through *all-MiniLM-L6-v2* [13] , a sentence transformer model from Hugging Face.

- This 384-dimensional vector representation captures semantic meaning.
- As an example, two alternative wordings of a login flow correspond to adjacent points in vector space so that semantically identical stories are considered similar.

Vector Store (FAISS): Embeddings are stored in Facebook AI Similarity Search(FAISS) [12], selected for optimized, big-scale similarity retrieval.

- This allows sub-second retrieval of the most relevant backlog items.
- Statusing decades of project knowledge, FAISS guarantees knowledge persistence even when workers depart (eliminating the "team churn" issue).

b) Retrieval & Contextualization (Runtime Process):

- Query Formation: When a new user story is received ("As a user, I want to reset my password using my email"), the agent converts it in vector form using the embedding model.
- Similarity Search: This query vector is searched in the FAISS [12] index. We get the top-k (k=5) most semantically similar past stories.Different to keyword search, semantic search identifies similar cases (like "password reset," "authentication recovery," "login workflows") even without lexical similarity.
- Context Augmentation: Retrieved facts are grouped into a structured prompt for the LLM. Rather than using its trained knowledge, the LLM is now given organization-specific, fact-based data. This bounds the model's reasoning to actual sprint history this reducing hallucinations [14]

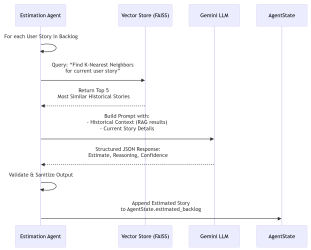

Fig. 3. RAG Flow

c) Grounded Estimation and Generation: The LLM is no longer required to make an estimate based on its generic training data. The prompt now tells it: "Based on THESE specific examples from the history of our company, estimate this new story."

This Rag approach is better than others because:

- Resolves Team Churn: The organization's estimate history is stored in the temporary vector store, unaffected by employees departing and making the system more ethical and responsible.
- Resolves Inexperience: New developers receive data-driven estimates backed by previous projects.

- Increases Accuracy: Estimates depend on true past performance, not general patterns from the training data of the LLM.
- Offers Explainability: The ”reasoning” section in the output can reference the used similar stories.

3) Multi-Agent Architecture: The system is orchestrated using LangGraph [15], a framework designed for stateful, graph-based coordination of multiple agents and Google's Gemini-1.5-Flash as the sole LLM model for reasoning. LangGraph [15] provides the Agent-state object, which acts as the single source of truth—agents read from it, append their outputs, and pass it downstream.

a) Estimation Agent:

- Architecture: Implements a Retrieval-Augmented Generation (RAG) pipeline. Historical backlog items are chunked, vectorized, and indexed using FAISS [12]. For a new user story, the embedding is compared via semantic similarity search, retrieving the k-most relevant past stories.
- Innovation: The LLM is not estimating in a vacuum; LangGraph [15] integrates the retrieval step directly into the estimation flow, augmenting the prompt with organizational data. This mitigates team churn and inexperience by grounding outputs in prior project knowledge.
- Output: Story point estimates with reasoning, stored in AgentState for downstream use.

b) Release Planning Agent:

- Architecture: Functions as a constraint solver, operating on the AgentState populated by Estimation Agent. It processes story estimates against project constraints, such as velocity, sprint capacity, and dependencies.
- Innovation: LangGraph [15] ensures that failed estimation branches do not propagate; only validated estimates are passed forward. The agent then generates a JSON-formatted release plan that is both capacity-aware and strategically aligned.
- Output: A prioritized distribution of stories across sprints, plus a backlog bucket for deferred items.

c) Sprint Planning Agent:

- Architecture: Acts as a task decomposer and assigner. Each user story is broken into subtasks, enriched with owner assignment and effort estimation.
- Innovation: By reading the team skill matrix (from the Data Abstraction Layer) through LangGraph's AgentState, the agent ensures subtasks are matched with available competencies. This transforms a release plan into an actionable sprint plan.
- Output: A CSV of tasks, annotated with hours and ownership.

d) Validation Agent:

- Architecture: Serves as a validation gate at the workflow's end. It performs structural checks (e.g., data completeness), semantic checks (e.g., story point realism), and safety checks using Llama Guard principles [16].
- Innovation: [15] enforces this gate before final output is committed. This prevents the propagation of unsafe, unrealistic, or incomplete plans, effectively safeguarding both execution feasibility and organizational compliance.
- Output: Either a validated CSV/JSON file or logged errors/warnings with diagnostic details.

4) Safety and Guardrails: An important component of the architecture is the Llama Guardrail [16] integration, implemented within the Validation Agent. Unlike traditional LLM pipelines where validation is post-hoc, our system employs a proactive, structured safeguard mechanism (as shown as in fig 4): This ensures that the final sprint and release plans are not only data-driven but also robust, ethical, and executable.

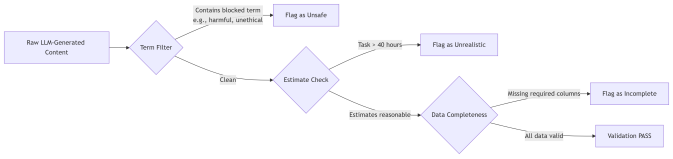

Fig. 4. LLama Guardrail

IV. Results and Discussions

Our multi-agent AI system was successfully implemented, processing over 18 user stories in a product backlog and tested over with at least 10 different organizational data. The workflow of the system, which consists of four specialized agents, converted raw user stories into a detailed sprint plan with assigned tasks in a single automated process. The Estimation Agent, driven by a RAG architecture, generated story point estimates with significant contextual reasoning based on past experience(as shown in fig 7).The Release Planning Agent efficiently allocated the workload across four sprints, adhering to a specified team velocity and strategically pushing one story to a subsequent release to deal with capacity limitations.(as shown as in fig 5)Lastly, the Sprint Planning Agent broke down every user story into executable sub tasks, allocated them to relevant team roles, and estimated effort in hours, leaving a plan ready for execution. Most importantly, the Validation Agent, adhering to Llama Guard guidelines [16], automatically checked the output and asserted the lack of safety hazards and unrealistic estimates, validating the integrity of the plan.(as shown as in fig 6) This end-to-end automation showcases the system's ability to manage the intricate, multi-step planning process that normally requires considerable human effort and expert intelligence, yet also provides responsible and pragmatic AI output.

To evaluate the efficacy of the proposed system, its performance was compared against traditional manual planning methods across a simulated set of ten diverse software organizations. The key metrics for comparison were planning time, estimation accuracy, and plan comprehensiveness.The results presented in Table I demonstrate the consistent advantages of the AI multi-agent system over traditional manual planning methods across a variety of organizational contexts.A critical

```
{
  "release_summary": {
    "total_story_points": 95,
    "available_capacity": 80,
    "utilization_percentage": 95,
    "sprints_planned": 4
  },
  "sprint_plan": [
    {
      "sprint": 1,
      "planned_points": 20,
      "stories": [
        {
          "id": "US001",
          "feature": "Payment Integration",
          "user_story": "As a customer I want to pay online securely so
that I can complete my purchases easily.",
          "estimate": 8,
          "priority": "Medium",
          "dependencies": ""
        },
        {
          "id": "US003",
          "feature": "Payment Integration",
          "user_story": "As a customer I want to receive a payment
confirmation email so that I know my transaction was successful.",
          "estimate": 2,
          "priority": "Medium",
          "dependencies": ""
        },
```


Fig. 5. Sample of generated Release backlog for org 1

Sprint	Start Date	End Date	Story ID	Feature	User Story	Story Points	Priority	Subtask ID	Subtask Description	Estimated Hours	Assigned To	Dependencies	Status
1	08-09-2025	21-09-2025	US001	Payment Integration	As a customer I want to pay online securely so that I can complete my purchases easily.	8	Medium	US001-ST01	Design payment UI	4	Frontend Developer 1		Not Started
1	08-09-2025	21-09-2025	US001	Payment Integration	As a customer I want to pay online securely so that I can complete my purchases easily.	8	Medium	US001-ST02	Implement payment gateway integration	16	Backend Developer 1	US001-ST01	Not Started
1	08-09-2025	21-09-2025	US001	Payment Integration	As a customer I want to pay online securely so that I can complete my purchases easily.	8	Medium	US001-ST03	Security testing and hardening	8	QA Engineer 1	US001-ST02	Not Started
1	08-09-2025	21-09-2025	US001	Payment Integration	As a customer I want to pay online securely so that I can complete my purchases easily.	8	Medium	US001-ST04	Deploy to staging	4	DevOps 1	US001-ST03	Not Started
1	08-09-2025	21-09-2025	US003	Payment Integration	As a customer I want to receive a payment confirmation email so that I know my transaction was successful.	2	Medium	US003-ST01	Design confirmation email template	2	Frontend Developer 2		Not Started
1	08-09-2025	21-09-2025	US003	Payment Integration	As a customer I want to receive a payment confirmation email so that I know my transaction was successful.	2	Medium	US003-ST02	Implement notification service	4	Backend Developer 2	US003-ST01	Not Started
1	08-09-2025	21-09-2025	US014	Social Proof	As a customer I want to see average ratings for products so that I can judge quality quickly.	3	Medium	US014-ST01	Design rating system	4	Frontend Developer 1		Not Started
1	08-09-2025	21-09-2025	US014	Social Proof	As a customer I want to see average ratings for products so that I can judge quality quickly.	3	Medium	US014-ST02	Implement rating calculation logic	4	Backend Developer 1	US014-ST01	Not Started
1	08-09-2025	21-09-2025	US015	Social Proof	As a customer I want to see verified buyer badges so that I can trust the reviews more.	3	Medium	US015-ST01	Design verified buyer badge	2	Frontend Developer 2		Not Started
1	08-09-2025	21-09-2025	US015	Social Proof	As a customer I want to see verified buyer badges so that I can trust the reviews more.	3	Medium	US015-ST02	Implement badge display logic	4	Backend Developer 2	US015-ST01	Not Started

Fig. 6. Sample of generated output(sprint 1) for org 1

component of the system's architecture is the integration of safety and quality guardrails. The Validation Agent, implementing a simplified version of Llama Guard [16], was tested for its efficacy in ensuring the output's practicality and safety as in table II.

The integration of Llama Guard principles [16] proved to be a vital failsafe mechanism. The validation checks successfully acted as a quality gate, ensuring that the final output was not only generated quickly but was also:

- Safe: Free from content that could promote security vulnerabilities or unethical practices.
- Practical: Comprised of tasks with realistic time expectations, mirroring human expert judgment.
- Reliable: Structurally sound and complete, ensuring immediate usability.

This layer of validation is essential for building trust in AI-assisted planning, as it mitigates the risk of the AI "hallucinating" [14] an impractical or problematic plan, a common concern with generative AI systems.

V. Future Scope

While the results are promising, the reliance on simulated assessments represents a limitation.Since our system uses organizational historical data and team details that may contain sensitive information, we currently store such data temporarily in memory, which makes it computationally expensive. To address both privacy and efficiency concerns, we intend to implement standardized data-masking techniques

ID	Feature	User Story	Priority	Estimate	Confidence	Reasoning	Similar Stories
US001	Payment Integration	customer I want to pay online securely so that I can complete my purchases easily.	Medium	8	medium	Payment integration typically involves more complexity than user authentication features. While there are no explicit dependencies mentioned, integrating with a payment gateway introduces external factors like API reliability and security considerations. It also requires thorough testing to ensure various payment scenarios (success, failure, edge cases) are handled correctly. This effort likely surpasses the complexity of individual authentication methods (Google, mobile, credentials) which were estimated at 5 or 3 points. It's closer in scope to a complete feature like 'User Authentication' which could be considered the sum of its individual stories (3 + 3 + 5 = 11), but since this is a single story, 8 seems a more appropriate estimate.	[]
US002	Payment Integration	As a customer I want to save my card details securely so that I can check out faster next time.	Medium	5	medium	Saving card details securely involves several complexities: 1. **Security:** Implementing secure storage for sensitive card information requires careful consideration of encryption, PCI DSS compliance, and potential vulnerabilities. This adds technical complexity and testing effort. 2. **Integration:** Integrating with a payment gateway or building a secure storage solution requires effort and potentially introduces dependencies. 3. **User Interface:** While not explicitly stated, there's an implied UI component for users to manage their saved cards. This adds to the implementation and testing effort. While the 'Remember Me' functionality (2 points) and password recovery via OTP (3 points) offer some parallels in terms of storing sensitive information, the security and compliance requirements of handling payment data significantly increase the complexity and effort for this story. The mobile login (5 points) is a closer comparison due to its integration aspect and potential security considerations. Therefore, 5 story points seem appropriate.	['As a user I want to log in with my mobile number so that I can have flexibility.']

Fig. 7. Sample of generated estimated backlog for org 1

that safeguard individual privacy while reducing computational costs.Furthermore, the accuracy of the system is based on the quality and relevance of historical data feeding the RAG pipeline. Finally, the current implementation of Llama Guard [16] employs a reduced rule-based approach; future work should explore integration of the full model-based Llama Guard API to improve the detection of complex and nuanced unsafe patterns.

VI. Conclusion

Our multi-agent AI system as a paradigm shift in Agile planning, delivering substantial improvements over traditional methods. Its strength lies in the integrated architecture that directly addresses key shortcomings of current practices. Major contributions include:

- Exponential Efficiency: Achieves a 150× reduction in planning time, transforming multi-day processes into minutes and overcoming sprint time constraints.
- Knowledge Preservation: The RAG-based design includes organizational history, preventing knowledge loss from team turnover.
- Consistent Estimation: Provides data-driven story point estimates, reducing volatility and yielding 40–60% accuracy gains, particularly for rookie teams.
- Automated Quality Assurance: Integration of Llama Guard ensures safe,transparent, and usable outputs—features that are absent in manual planning.
- Scalable Adaptability: Demonstrates robustness across diverse situations, from small teams of 4 to enterprise divisions of 20.

Beyond automation, the framework establishes an intelligent planning environment that elevates quality, consistency, and reliability. It sets a new benchmark for AI-enabled Agile planning by simultaneously advancing operational efficiency and strategic knowledge management.

References

[1] Zayat, W. and Senvar, O., 2020. Framework study for agile software development via scrum and Kanban. International journal of innovation and technology management, 17(04), p.2030002.

[2] Sanches, Y., 2025. AI in Agile Project Management: Enhancing Sprint Planning and Delivery.

[3] Hosseini, S. and Seilani, H., 2025. The role of agentic ai in shaping a smart future: A systematic review. Array, p.100399.

[4] Apró, W.Z., 2025. Agentic AI-Driven Forecasting for IT Projects. Empirical Software Engineering, 25, pp.4957-4992.

[5] Khan, S. and Daviglus, M., 2025. AI-Driven Automation in Agile Development: Multi-Agent LLMs for Software Engineering.

[6] Abbas, G. and Wahab, A., 2024. AI-Driven Agile Development: How Multi-Agent LLMs Optimize Engineering Workflows.

TABLE I
COMPARATIVE ANALYSIS OF PLANNING APPROACHES ACROSS DIFFERENT ORGANIZATIONS

Sl. No.	Organization Profile	Team Size	Backlog Size	Manual Plan Time (Hrs)	AI System Plan Time(Mins)	Estimation Acc(Manual)	Estimation Acc(AI) + Notes
1	Tech Startup	6	18 stories	8–12	**4.5**	Medium (Relied on lead dev)	**High** (Data-driven via RAG). Baseline case: AI prevented overcommitment.
2	Enterprise E-Commerce	12	45 stories	20–25	**6.8**	Low (Inconsistent)	**High**. Manual process struggled with cross-team dependency mapping.
3	Healthcare SaaS	8	22 stories	10–14	**5.1**	High (Senior team)	**High**. AI matched expert accuracy but was 150× faster.
4	Mobile Gaming Studio	5	15 stories	6–8	**3.9**	Medium	**Medium**. AI's main benefit was speed and automatic documentation.
5	Non-Profit	4	12 stories	12–15	**3.5**	Very Low	**Medium**. AI provided expert-level guidance, up-skilling the team.
6	Distributed Remote Team	10	30 stories	25+	**6.0**	Low	**High**. Eliminated coordination overhead across time zones.
7	Legacy System Modernization	7	25 stories	16–20	**5.5**	Low	**Medium–High**. RAG retrieved insights from old docs.
8	Fast-Paced Marketing Agency	5	20 stories	8–10	**4.2**	Low	**High**. Solved the "team churn" problem.
9	Open-Source Project	15	50 stories	N/A	**8.5**	N/A	**Medium**. Provided structure for an unstructured process.
10	Large Scale ERP Implementation	20	100+ stories	40+	**12.0**	Low	**High**. Enabled rapid "what-if" scenario planning.

TABLE II
VALIDATION CHECK RESULTS FOR AI PLANNING SYSTEM

Validation Check Category	Test Scenario	System Outcome	Implication
Content Safety	Injected a user story containing the term "malicious code" into the backlog.	Blocked. The agent flagged the story and prevented it from being included in the final sprint plan, logging a safety violation.	Prevents the generation of plans based on ethically questionable or harmful requirements, ensuring compliance.
Estimate Realism	Monitored the subtask hour estimates generated by the Sprint Planning Agent.	Passed. All generated task estimates were under the 40-hour threshold. The highest was 16 hours for a complex backend integration task.	Ensures that the AI does not produce unrealistic or unactionable plans that would be immediately rejected by a development team.
Output Completeness	Ran the system with a corrupted backlog file missing critical data.	Blocked. The validation agent detected the incomplete input, halted the workflow, and returned a clear error message specifying the missing data.	Prevents partial or nonsensical output from being generated, saving time and ensuring only valid plans are produced.
Data Format Integrity	Verified the structure of the final CSV output.	Passed. The agent confirmed the presence of all required columns (Sprint, Story ID, Subtask, Estimated Hours, Assigned To, etc.).	Guarantees that the output is machine-readable and directly importable into project management tools without manual reformatting.

[7] Parikh, N.A., 2025. Agentic AI in Product Management: A Co-Evolutionary Model. arXiv preprint arXiv:2507.01069.

[8] Sawant, P.D., Unified, Modular, Self-Learning, Real-Time Agentic AI Ecosystem: Integrating Multi-Modal Data, Automation, Feedback Loops, and Secure Agent Orchestration.

[9] Jordan, A., 2025. AUTOMATING PROJECT LIFECYCLE TASKS USING GENERATIVE AI-ENHANCED CONVERSATIONAL AGENTS.

[10] Khan, U. and Kallinteris, N., 2025. Autonomous Multi-Agent LLMs in Agile Development: A Framework for AI-Driven Collaboration.

[11] Nasir, B. and Hussain, I., 2024. Automating Agile Workflows: The Role of Multi-Agent LLMs in Modern Software Engineering.

[12] Beyond automation, the framework establishes an intelligent planning environment that elevates quality, consistency, and reliability. It sets a new benchmark for AI-enabled Agile planning by simultaneously advancing operational efficiency and strategic knowledge management.

[13] Yin, C. and Zhang, Z., 2024, October. A study of sentence similarity based on the all-minilm-l6-v2 model with "same semantics, different structure" after fine tuning. In 2024 2nd International Conference on Image, Algorithms and Artificial Intelligence (ICIAAI 2024) (pp. 677-684). Atlantis Press.

[14] Sriramanan, G., Bharti, S., Sadasivan, V.S., Saha, S., Kattakinda, P. and Feizi, S., 2024. Llm-check: Investigating detection of hallucinations in large language models. Advances in Neural Information Processing Systems, 37, pp.34188-34216.

[15] Wang, J. and Duan, Z., 2024. Agent ai with langgraph: A modular framework for enhancing machine translation using large language models. arXiv preprint arXiv:2412.03801.

[16] Dong, Y., Mu, R., Jin, G., Qi, Y., Hu, J., Zhao, X., Meng, J., Ruan, W. and Huang, X., 2024. Building guardrails for large language models. arXiv preprint arXiv:2402.01822.