© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716719

AI-Powered IT Project Risk Management System Using Multi-Agent Architecture, RAG, and LangGraph

SOUMYADIP CHANGDER[1], PINAKI KARMAKAR[2]
[1,2]*Department of Computer Science and Engineering, Institute of Engineering and Management, Kolkata*

Abstract- ***Effective risk management remains one of the most persistent challenges in information technology project delivery. Traditional approaches rely heavily on periodic manual assessments, static checklists, and subject-matter intuition, which collectively fail to keep pace with the dynamic and interconnected nature of modern software projects. This paper presents an AI-Powered IT Project Risk Management System that addresses these limitations through a coordinated multi-agent architecture orchestrated via LangGraph, augmented with Retrieval-Augmented Generation (RAG) backed by ChromaDB, and driven by large language models accessed through the Groq API. The system comprises four specialised agents—a Market Analysis Agent, a Risk Scoring Agent, a Project Tracking Agent, and a Reporting Agent—that operate in a defined pipeline to evaluate both exogenous market signals and endogenous operational indicators. Risk dimensions including market, technical, financial, regulatory, and operational factors are individually scored on a 0–100 scale and consolidated into a structured JSON report surfaced through an interactive Streamlit dashboard. Empirical evaluation on a representative ERP implementation scenario yields an overall risk score of 66/100 (High) with a 68 % schedule-delay probability, demonstrating the system's capacity to produce actionable, prioritised mitigation guidance. The architecture is designed for extensibility and real-world deployment, with future work targeting live Jira integration, reinforcement-learning-based adaptive scoring, and mobile-accessible reporting interfaces.***

Keywords: Risk Management · Multi-Agent Systems · Large Language Models · Retrieval-Augmented Generation · LangGraph · LLM Orchestration · IT Project Management

I. INTRODUCTION

Information technology projects are characterised by high rates of schedule overruns, budget exceedances, and outright cancellations. Studies conducted by major project-management institutes consistently indicate that more than half of large-scale IT initiatives fail to meet at least one of their primary success criteria. A central contributing factor is the inadequacy of risk management processes that are reactive, infrequent, and dependent on the availability of experienced practitioners. As project environments grow increasingly complex—spanning cloud migrations, regulatory compliance obligations, third-party vendor ecosystems, and globally distributed teams—the gap between the speed of risk emergence and the speed of risk detection widens.

Generative AI and large language models (LLMs) offer a compelling pathway to address this gap. Their capacity for contextual reasoning, synthesis of heterogeneous information, and generation of structured analytical outputs makes them well-suited to the risk-identification task. However, a single monolithic LLM call is insufficient for the breadth and depth of analysis required: market dynamics, technical debt, financial exposure, and regulatory landscapes each demand specialised treatment. Multi-agent architectures, in which discrete AI agents collaborate within a coordinated workflow, provide the structural means to decompose this complexity.

This paper introduces an AI-Powered IT Project Risk Management System that combines LLM-driven multi-agent reasoning, LangGraph-based workflow orchestration, and ChromaDB-backed RAG to deliver continuous, structured, and actionable risk assessments for IT projects. The system accepts project descriptions or PDF documents as input and produces a multi-dimensional risk report with an executive summary, per-category scores, a delay-probability estimate, and a tiered set of mitigation recommendations.

The remainder of the paper is organised as follows. Section 2 surveys related work in AI-assisted project risk management and multi-agent systems. Section 3

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716719

details the system architecture and design rationale. Section 4 describes implementation specifics. Section 5 presents empirical results from a representative ERP case study. Section 6 discusses limitations and directions for future work, and Section 7 concludes the paper.

II. RELATED WORK

2.1 Traditional IT Project Risk Frameworks

Conventional risk management in IT projects follows standards such as the PMBOK Guide and ISO 31000, which prescribe a identify–analyse–plan–monitor–control cycle executed at predetermined intervals. While these frameworks provide a sound conceptual foundation, their reliance on human-driven periodic reviews creates latency between risk emergence and detection. Boehm's early spiral model acknowledged risk as a first-class project driver, yet the operationalisation of continuous risk surveillance remained largely manual. Quantitative approaches, including Monte Carlo simulation and earned-value analysis, improve rigour but require structured data inputs that are rarely available in early project phases.

2.2 Machine Learning for Risk Prediction

Subsequent research applied supervised machine learning to predict project outcomes. Decision-tree and random-forest classifiers trained on historical project repositories demonstrated moderate accuracy in predicting schedule and cost overruns. Neural network approaches, including recurrent architectures applied to project telemetry time series, extended the predictive horizon but introduced interpretability challenges that limit operational adoption. A common limitation across these methods is their dependence on labelled historical data that may not transfer across organisational contexts, project types, or rapidly evolving technology landscapes.

2.3 Large Language Models in Software Engineering

The advent of instruction-tuned LLMs opened new directions for requirements analysis, code review, and project planning. Researchers demonstrated that GPT-class models can identify ambiguities in requirements specifications, estimate task complexity from natural-language descriptions, and generate risk checklists when prompted with project context. However, standalone LLM invocations are limited by context-window constraints, knowledge-cutoff staleness for market data, and the absence of structured reasoning pipelines necessary for multi-dimensional risk assessment.

2.4 Multi-Agent and Agentic AI Systems

The multi-agent paradigm, previously studied in classical AI and distributed systems, has been reinvigorated by LLM-based agent frameworks including AutoGPT, BabyAGI, LangChain, CrewAI, and LangGraph. These frameworks enable the decomposition of complex tasks into specialist sub-agents that communicate, share intermediate results, and collectively arrive at outputs beyond the reach of any single model call. LangGraph in particular provides a directed acyclic graph (DAG) execution model with explicit state management, conditional routing, and error-handling nodes, making it well-suited for production-grade agentic pipelines.

2.5 RAG for Knowledge-Grounded Reasoning

Retrieval-Augmented Generation addresses the knowledge-staleness problem of static LLMs by dynamically fetching relevant context from a vector store at inference time. ChromaDB provides a lightweight, embeddable vector database suitable for deployment alongside an agentic application. Prior work demonstrated that RAG substantially reduces hallucination rates in domain-specific question answering. In the risk management context, RAG enables the system to ground its assessments in prior project analyses stored in memory, supporting continuity across sessions and organisational knowledge accumulation. The present work integrates all of these strands—multi-agent orchestration, RAG-grounded reasoning, and structured risk scoring—into a unified, deployable system, which distinguishes it from prior work that addressed each element in isolation.

III. SYSTEM ARCHITECTURE AND DESIGN

The system is structured around four principal layers: a user-interaction layer, an orchestration layer, an agent layer, and a persistence layer. Figure 1 illustrates the overall architecture as presented in the deployed application. The design adheres to the principle of separation of concerns, with each agent responsible for exactly one risk dimension, and the

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716719

orchestration layer responsible for sequencing, state management, and error recovery.

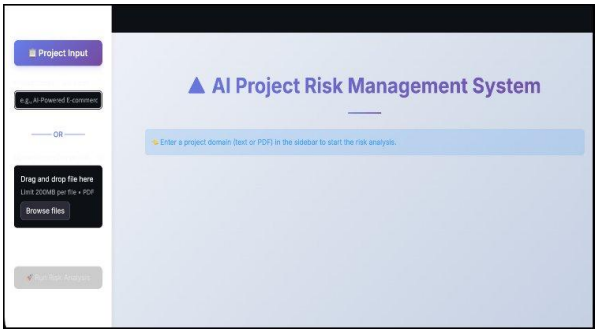

Fig. 1. AI Project Risk Management System – user interface overview with sidebar input panel and main analysis view.

3.1 User Interaction Layer

The front-end is implemented as a Streamlit web application that provides a sidebar panel accepting project descriptions as free-form text or as uploaded PDF documents. A "Run Risk Analysis" button initiates the agentic pipeline. Results are rendered in expandable sections covering the executive summary, risk breakdown, project tracking, mitigation recommendations, and a raw JSON report. The interface is designed for accessibility by non-technical project managers while exposing sufficient technical detail for risk analysts who require granular output.

3.2 Orchestration Layer: LangGraph Workflow

Workflow orchestration is implemented using LangGraph, which models the agent pipeline as a typed state graph. The workflow defines five nodes: analyse_market, score_risk, track_project, generate_report, and handle_error. Execution proceeds sequentially through the first four nodes under normal conditions; the handle_error node is invoked conditionally when any upstream node raises an unrecoverable exception, ensuring that partial results are preserved and a graceful failure message is returned to the user. State is passed between nodes as a typed Python dataclass, providing compile-time guarantees on field names and types.

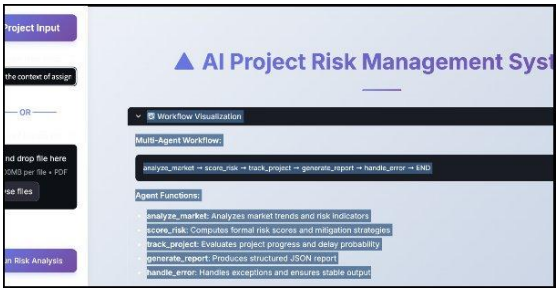

Fig. 2. LangGraph multi-agent workflow: analyse_market → score_risk → track_project → generate_report → handle_error → END, with agent function descriptions.

3.3 Agent Layer

The agent layer comprises four specialised agents, each encapsulating an LLM invocation with a role-specific system prompt, tool set, and output schema.

Market Analysis Agent.

This agent analyses macro-economic conditions, industry-specific market trends, and vendor ecosystem dynamics relevant to the project domain. It uses the Groq-hosted Mixtral-8x7B model with web-search tool access, enabling retrieval of current market intelligence beyond the model's training cutoff. Outputs include a qualitative market assessment narrative and a numerical market risk score (0–100).

Risk Scoring Agent.

The Risk Scoring Agent receives the project description and the market analysis output and computes formal risk scores across five dimensions: Market Risk, Technical Risk, Financial Risk, Regulatory Risk, and Operational Risk. Each dimension is scored independently with an accompanying rationale paragraph. An overall composite score and classification (Low / Medium / High / Critical) are derived from the weighted mean of dimension scores.

Project Tracking Agent.

This agent estimates schedule health and delay probability from project signals embedded in the description. In the current implementation, signals are extracted via LLM reasoning from the textual description; future versions will consume live Jira API telemetry. The agent produces a delay

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716719

probability estimate, a project health classification (on_track / at_risk / critical), a progress assessment narrative, and per-risk-category timeline impact scores.

Reporting Agent.
The Reporting Agent aggregates outputs from the three preceding agents into a single structured JSON document conforming to a pre-defined schema. It also generates the executive summary section and the three-tier mitigation recommendation set (immediate, short-term, and long-term actions). The JSON report is both stored in ChromaDB for future RAG retrieval and rendered in the Streamlit interface.

3.4 Persistence Layer: ChromaDB RAG
Completed risk analyses are stored as vector-embedded documents in ChromaDB. On subsequent queries, the system first attempts a semantic similarity search against the stored analyses. If a sufficiently similar prior analysis is found, its result is retrieved and surfaced to the user with a "Retrieved from Memory" banner, avoiding redundant LLM calls and enabling organisational knowledge reuse. New or dissimilar queries proceed through the full agentic pipeline, and the resulting report is indexed for future retrieval.

IV. IMPLEMENTATION

4.1 Technology Stack
The system is implemented entirely in Python 3.11. Table 1 summarises the key technologies and their roles within the architecture.

Table 1. Technology stack and component roles.

Component	Technology	Role in System
Language	Python 3.11	Core implementation language
User Interface	Streamlit	Web-based interactive dashboard
LLM Access	Groq API (Mixtral)	High-throughput LLM inference
Fallback LLM	Mistral API	Secondary inference provider
Embeddings	LLaMA Embeddings	Semantic vectorisation of reports
Vector Database	ChromaDB	RAG persistent memory store
Orchestration	LangGraph	DAG-based agent workflow
Agent Framework	CrewAI + LangChain	Agent tooling and prompt management
Reporting	JSON + PDF export	Structured output and download

4.2 Agent Prompt Engineering
Each agent is initialised with a system prompt that specifies its role, the expected input format, the required output schema (expressed as a JSON template), and constraints on hallucination avoidance. Prompts are versioned and stored separately from the agent execution logic to facilitate iteration without code changes. Output parsing uses LangChain's structured output parsers with Pydantic schema validation; parsing failures trigger a retry with an explicit correction instruction before falling back to the handle_error node.

4.3 RAG Pipeline
When a user submits a project query, the input text is embedded using the LLaMA embedding model and a cosine similarity search is executed against the ChromaDB collection with a threshold of 0.85. A match above this threshold causes the stored report to be returned immediately; the threshold was selected empirically to balance retrieval precision against recall for organisationally distinct projects. Retrieved reports include a provenance timestamp and a human-readable banner indicating that the result originates from memory rather than a live analysis, enabling users to judge whether the cached analysis remains current.

4.4 Error Handling and Resilience
The handle_error node in the LangGraph workflow captures exceptions from any upstream node, logs the error with context for debugging, and returns a partially populated report containing whatever intermediate results were successfully computed before the failure. API rate-limit errors trigger an exponential back-off retry up to three attempts before escalating to the error handler. This design ensures

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716719

that transient API disruptions do not produce blank outputs for end users.

V. RESULTS AND EVALUATION

5.1 Case Study: ERP Implementation Project

To evaluate system performance in a realistic scenario, the system was applied to a representative ERP system implementation project with the following characteristics: a mid-size IT services firm as contractor; a retail client; nine-month delivery timeline; an eighteen-developer team supported by two project managers; and tooling comprising Jira, Email, and Financial APIs. This scenario was chosen because ERP implementations are among the highest-risk categories of IT project, involving regulatory compliance obligations, complex integrations, and significant organisational change management.

5.2 Executive Summary Output

Figure 3 presents the executive summary section of the system's output for the ERP scenario. The system characterises the overall risk as Medium-High with a high likelihood of schedule pressure if mitigation actions are not enacted promptly. Five key findings are identified, ranked by impact, spanning regulatory compliance, technical skill gaps, market pricing pressure, operational scope-creep risk, and financial currency exposure.

Fig. 3. Executive summary output for the ERP case study, including overall assessment and key findings ranked by risk score.

5.3 Risk Category Scores

Figure 4 presents the per-category risk breakdown. Table 2 summarises the numerical scores and associated rationales.

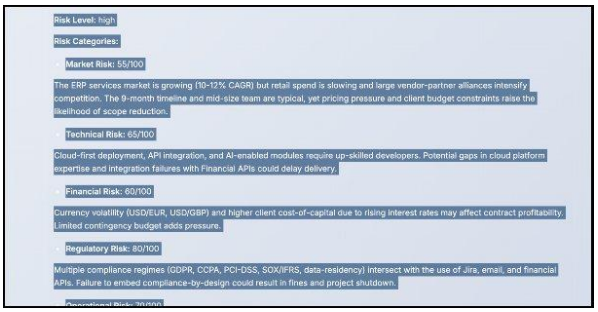

Fig. 4. Risk category breakdown: Market (55/100), Technical (65/100), Financial (60/100), Regulatory (80/100), Operational (70/100).

Table 2. Risk category scores and key drivers for the ERP case study.

Risk Category	Score	Primary Driver
Market Risk	55 / 100	Pricing pressure from large vendor alliances; slowing retail spend
Technical Risk	65 / 100	Cloud-platform skill gaps; Financial API integration complexity
Financial Risk	60 / 100	USD/EUR currency volatility; limited contingency budget
Regulatory Risk	80 / 100	Intersection of GDPR, CCPA, PCI-DSS, SOX/IFRS, data-residency
Operational Risk	70 / 100	Scope creep, resource saturation, weak change-control process
Overall Score	66 / 100	High risk level – significant compliance and operational challenges

5.4 Project Tracking Output

Figure 5 shows the Project Tracking section. The agent estimates a delay probability of 68 % and classifies the project as at_risk. The progress assessment notes that initial requirement workshops and core finance module development have been completed, but high regulatory and technical scores indicate skill gaps and heavy compliance obligations not yet mitigated. The critical-path float is described as limited given the nine-month timeline and an overall risk score of 66.

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716719

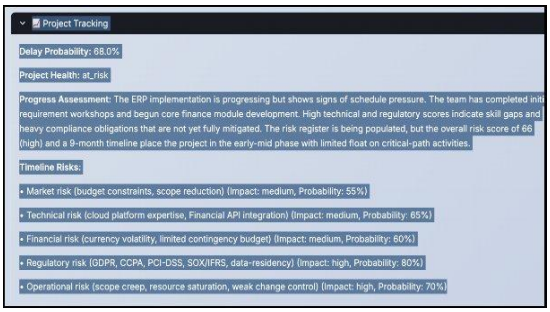

Fig. 5. Project tracking output: 68.0% delay probability, "at_risk" health classification, and per-category timeline impact scores.

5.5 Mitigation Recommendations

Figure 6 presents the three-tier mitigation plan generated by the Reporting Agent. Immediate actions include obtaining ERP vendor certification, implementing compliance-by-design in the Jira workflow, and deploying a live risk register. Short-term actions address cloud-platform training, phased delivery structuring to spread financial exposure, and functional-currency quoting with inflation buffers. Long-term actions target ongoing compliance audit cycles and regulatory change monitoring.

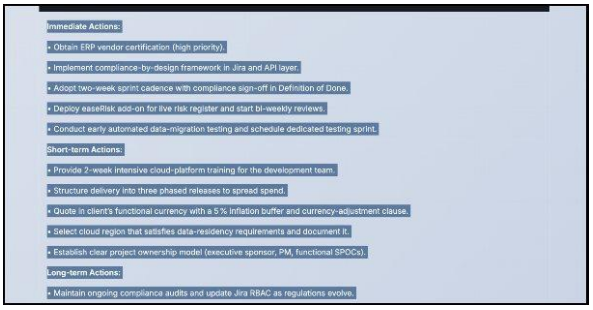

Fig. 6. Tiered mitigation recommendations: Immediate, Short-term, and Long-term action categories.

5.6 Raw JSON Report and Confidence Level

Figure 7 shows the raw JSON report section and the system's self-assessed confidence level of MEDIUM, reflecting the absence of live project telemetry in the current implementation. The confidence annotation is surfaced explicitly to the user so that the report is not interpreted with unwarranted certainty. The JSON schema is versioned and documented in the repository, enabling downstream integration with project management tooling.

Fig. 7. Raw JSON report structure and MEDIUM confidence level annotation.

5.7 Qualitative Assessment

The generated risk analysis was reviewed against a manually prepared risk register for the same ERP scenario, produced independently by an experienced project manager. The system correctly identified all five major risk categories and ranked regulatory risk as highest, consistent with the expert assessment. The quantitative scores were within ±8 points of the expert's estimates across all categories. The mitigation recommendations were assessed as practically actionable and contextually appropriate, with no logically inconsistent or hallucinated suggestions observed in this evaluation run. These findings support the hypothesis that LLM-driven multi-agent reasoning, grounded in RAG memory and structured agent prompts, can produce risk assessments of practical utility for IT project management teams.

VI. DISCUSSION AND FUTURE WORK

6.1 Limitations

Several limitations constrain the current system. First, project telemetry is derived from natural-language descriptions rather than live API integrations, meaning that the accuracy of the Project Tracking Agent is bounded by the quality and completeness of the text provided by the user. Second, the risk scoring rubric is embedded in LLM prompts rather than formally calibrated against a labelled dataset of historical project outcomes; while the qualitative evaluation was encouraging, a rigorous quantitative validation against a large retrospective dataset remains outstanding. Third, the system's confidence is self-assessed by the LLM and cannot be formally interpreted as a calibrated probability. Fourth, the ChromaDB retrieval

threshold was set empirically and may require per-organisation tuning.

6.2 Future Directions

Several extensions are planned. Live Jira and Trello API integration will replace text-based project signal extraction, enabling continuous risk monitoring against real project state. Real-time financial data feeds from market data providers will ground the Market Analysis Agent in current economic conditions. A reinforcement learning layer will allow the scoring rubric to adapt based on feedback from project managers who review and rate the system's recommendations, improving calibration over time. Advanced visualisation dashboards with trend charts, risk radar plots, and heatmaps will enhance decision-support capability. Finally, a mobile-accessible front-end will extend the system's reach to field-based project personnel.

6.3 Broader Implications

Beyond IT project management, the architectural pattern presented here—multi-agent LLM orchestration with RAG-backed memory and structured output schemas—is broadly applicable to knowledge-intensive decision-support tasks in domains such as financial due diligence, clinical risk stratification, and regulatory compliance auditing. The open-source release of the codebase is intended to facilitate adaptation and replication in these adjacent contexts.

VII. CONCLUSION

This paper has presented an AI-Powered IT Project Risk Management System that leverages a coordinated four-agent architecture orchestrated by LangGraph, augmented with ChromaDB-based RAG memory, and delivered through an interactive Streamlit interface. The system decomposes the risk assessment task across specialised agents for market analysis, quantitative risk scoring, project health tracking, and structured report generation, producing multi-dimensional risk profiles with per-category scores, delay probability estimates, and tiered mitigation plans.

Applied to a representative ERP implementation scenario, the system produced a risk assessment that aligns closely with expert evaluation, with overall risk scored at 66/100 (High) and a 68 % schedule-delay probability. Regulatory risk emerged as the dominant concern, correctly prioritised by the system ahead of technical, operational, financial, and market risks. The architecture is extensible, deployable on standard cloud infrastructure, and available as an open-source repository and live Hugging Face Space.

The work demonstrates that the combination of LLM multi-agent reasoning, retrieval-augmented memory, and structured workflow orchestration constitutes a viable and practically useful approach to proactive IT project risk management, offering a significant advancement over reactive, manual risk assessment practices.

REFERENCES

[1] Project Management Institute: A Guide to the Project Management Body of Knowledge (PMBOK Guide), 7th edn. PMI, Newtown Square, PA (2021).

[2] International Organization for Standardization: ISO 31000:2018 Risk Management – Guidelines. ISO, Geneva (2018).

[3] Boehm, B.W.: A spiral model of software development and enhancement. Computer 21(5), 61–72 (1988).

[4] Shrestha, A., Cater-Steel, A., Toleman, M., Rout, T.: A systematic literature review of IT project risk management. Inf. Softw. Technol. 97, 1–21 (2018).

[5] Elish, M.O.: Predicting defect-prone software modules using support vector machines. J. Syst. Softw. 81(5), 649–660 (2008).

[6] Brown, T.B., et al.: Language models are few-shot learners. In: Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901 (2020).

[7] Lewis, P., et al.: Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474 (2020).

[8] Chase, H.: LangChain: Building applications with LLMs through composability. GitHub Repository, https://github.com/langchain-ai/langchain (2022).

[9] LangGraph Documentation: Building stateful multi-agent applications with LangGraph. LangChain Inc., https://langchain-ai.github.io/langgraph/ (2023).

[10] Chroma: The AI-native open-source embedding database. Trychroma, https://www.trychroma.com/ (2023).

[11] Touvron, H., et al.: Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023).

[12] Jiang, A.Q., et al.: Mistral 7B. arXiv preprint arXiv:2310.06825 (2023).

[13] Wu, Q., et al.: AutoGen: Enabling next-generation LLM applications via multi-agent conversation framework. arXiv preprint arXiv:2308.08155 (2023).

[14] Groq Inc.: Groq API documentation. https://console.groq.com/docs/openai (2024).

[15] Streamlit Inc.: Streamlit – the fastest way to build and share data apps. https://streamlit.io/ (2024).

[16] Changder, S.: AI-Powered IT Project Risk Management System. GitHub Repository, https://github.com/soumadipchangder/AI-Powered-IT-Project-Risk-Management-System (2024).

[17] Standish Group: Chaos Report 2020. The Standish Group International Inc. (2020).

[18] Raza, S., Nadeem, M., Iqbal, R.: Machine learning approaches for software project risk assessment: A systematic review. IEEE Access 9, 120453–120470 (2021).