



Integrating Traditional Risk Management Practices with Agile Software Development: A Hybrid Framework for Improved Risk Handling

¹Nidheesh Sharma, ²Dr.Neha Gupta

¹Research scholar, ²Associate Professor

Engineering - Application, Department of Computer Science and Engineering

¹nidheeshsharma.10@gmail.com, ²nehaguptanneelkanth@gmail.com

<https://doi.org/10.64882/ijrt.v14.i3.1874>

ABSTRACT

Software projects continue to overrun schedule and budget for reasons that disciplined risk management was designed to prevent. Plan-driven risk management, as codified in PMBOK, PRINCE2 and ISO 31000, offers rigorous identification, quantification and traceability, but its heavyweight documentation cycles fit poorly with short iterations and emergent requirements. Agile methods handle volatility well through short feedback loops, yet they treat risk implicitly, rarely quantify exposure, and consistently underperform on slow-moving systemic risks such as architectural erosion, vendor dependency and regulatory compliance. This paper proposes HARM-F (Hybrid Agile Risk Management Framework), a two-tier framework that retains a lightweight strategic risk register and quantitative exposure model from traditional practice while embedding risk activities directly in agile ceremonies at the tactical tier. The framework introduces a composite Hybrid Risk Score that combines probability, impact, volatility and detectability with analytic-hierarchy-process weights, a risk-velocity measure that converts the static register into an iteration-level trend signal, and a benefit-ratio rule for selecting responses. HARM-F was developed through a design-science cycle informed by a practitioner survey and evaluated on six industrial projects against matched control projects. The treatment group showed a 37.9 percent gain in risk-detection effectiveness, a 60.8 percent reduction in escaped risks, and schedule and cost variance reduced to 7.4 and 6.8 percent respectively, at an additional process overhead of roughly four percent of iteration effort.

Keywords: software risk management; agile software development; hybrid framework; risk exposure; Scrum; design science; project governance

1. INTRODUCTION

1.1 Background of the Study

Risk management became a formal concern in software engineering when it was recognised that most project failures are not caused by unknown technology but by known threats that were never tracked to closure. Classical treatments framed risk as a quantity to be identified, analysed, prioritised and controlled through an explicit, repeatable process, and this view was institutionalised in the process standards that still dominate governance practice: PMBOK's risk knowledge area [3], PRINCE2's risk theme, ISO 31000 [4] and the risk process areas of



CMMI. These approaches share three assumptions: that the scope of the project is largely knowable in advance, that risks can be enumerated up front in a register, and that the register is reviewed on a fixed managerial cadence, typically monthly or at phase gates.

Agile software development dissolved those assumptions. Scrum [5], Kanban and scaled variants such as SAFe and LeSS [9] replaced up-front specification with incremental delivery, working software and continuous customer collaboration. Their designers argued that agility is itself a risk-control strategy: short iterations limit the amount of work exposed to a change in requirements, continuous integration bounds the cost of defect discovery, and frequent demonstrations reduce the probability of building the wrong product. Empirically, this argument holds for requirement volatility and for certain classes of technical uncertainty [2], [6]. It holds less well elsewhere. Because agile teams manage risk implicitly, through the backlog and through informal conversation, risks that do not surface within a single iteration horizon tend to remain invisible until they are expensive. Architectural erosion, security and regulatory exposure, key-person dependency, supplier failure and integration debt all accumulate on timescales far longer than a two-week sprint [16], [17].

The resulting situation is not a choice between two philosophies but a coverage problem. Plan-driven governance sees the slow risks and misses the fast ones; agile practice sees the fast risks and misses the slow ones. Most organisations, particularly in regulated or contractually bound domains, already operate somewhere between the two, but they do so informally, with no defined interface between the corporate risk register and the sprint board.

1.2 Scope and Motivation

This study is scoped to iterative, team-level software delivery in organisations that carry an external governance obligation, such as contractual commitments, audit requirements or regulatory oversight. It addresses risk handling from identification through monitoring and closure within a single delivery programme; it does not address enterprise portfolio risk, financial hedging or organisational change management. The motivation is practical: practitioners repeatedly report [19] that they are required to maintain a formal register for governance while running the actual project from a board that does not reference it, producing duplicated effort and two divergent views of the same project.

1.3 Research Objectives

The study pursues two objectives:

- To design a hybrid risk management framework that integrates the quantitative rigour and traceability of traditional risk management with the cadence and feedback mechanisms of agile software development, without exceeding the process overhead tolerance of an agile team.
- To evaluate the proposed framework empirically against purely traditional and purely agile practice using defined performance metrics for risk detection, risk escape, response latency, and schedule and cost variance.

1.4 Significance of the Study

The contribution is threefold. For practice, the framework offers a deployable two-tier structure with defined artefacts, roles and cadences that can be adopted incrementally by a



Scrum team without abandoning existing governance instruments. For measurement, it supplies a composite risk score and a risk-velocity indicator that make risk trend visible on the same timescale as the burndown chart, converting risk from a periodic report into a continuously observable project variable. For research, it provides an evaluated design artefact and a metric set that subsequent studies can replicate, addressing the recurring criticism that hybrid risk management is discussed anecdotally rather than measured.

2. LITERATURE REVIEW

2.1 Traditional Risk Management in Software Projects

The traditional canon models risk as the product of the probability of an undesirable event and its impact, an exposure value used to rank a register of enumerated threats [3], [4]. Around this core sit structured identification techniques such as checklists, taxonomy-based questionnaires, failure mode and effects analysis and structured brainstorming; quantitative analysis techniques including expected monetary value, decision trees, sensitivity analysis and Monte Carlo simulation over schedule networks; and response planning categorised as avoid, transfer, mitigate or accept. The strengths of this body of work are well documented: it produces an explicit, auditable artefact, it forces early consideration of low-probability high-impact events, and it supports contractual and insurance decisions that require numbers rather than opinions.

Its weaknesses are equally documented. Register maintenance is effort-intensive and tends to decay after the planning phase; probability estimates are elicited once and rarely revised; and the review cadence is tied to management reporting rather than to the rate at which the project actually changes. Studies of register quality repeatedly find [1] that a large share of entries are never updated after creation, which means the artefact reflects the project as imagined at kickoff rather than as it exists.

2.2 Agile Software Development and Risk Management

Agile literature treats risk largely as an emergent property of the process rather than as an object of explicit management. Iteration planning limits work in progress and therefore exposure; the definition of done constrains quality risk; retrospectives surface process risk; continuous integration and automated testing compress the feedback loop within which defects can be detected. Several authors describe this as implicit risk management [6] and argue that it is sufficient for the majority of project uncertainty. Empirical work supports the claim for requirement and design volatility, where agile teams demonstrably outperform plan-driven teams on responsiveness [2], [16].

The counter-evidence concerns persistence and quantification. Because risks are represented as backlog items, spikes or impediments, they inherit the lifespan of a sprint: an impediment that cannot be resolved within the iteration is often re-raised, deprioritised and eventually normalised. There is no exposure value, so competing risks are ordered by advocacy rather than by magnitude, and there is no trend data, so a team cannot tell whether its overall risk position is improving. Scaled frameworks partially address this through program-level risk boards and PI-planning ROAM classification [9], but these are coarse and operate at a

cadence of two to three months, which reintroduces the latency problem that agility was meant to remove.

2.3 Comparison of Traditional and Agile Approaches

Table 1 contrasts the two paradigms along the dimensions relevant to framework design [2], [16]. The comparison shows that the approaches are not opposed on principle but complementary in coverage: each is strong precisely where the other is weak, which is the structural argument for integration rather than selection.

Table 1. Dimensional comparison of traditional and agile risk management

Dimension	Traditional (plan-driven)	Agile (iterative)
Risk visibility	Explicit, centralised register	Implicit, embedded in backlog and impediments
Identification timing	Front-loaded at planning and phase gates	Continuous but unstructured and undocumented
Quantification	Probability $\times$ impact, EMV, simulation	Largely absent; ordinal or verbal judgement
Review cadence	Monthly or phase-gate (4–12 weeks)	Daily and per-iteration (1–4 weeks)
Ownership	Project manager / risk officer	Whole team; no designated accountability
Response planning	Formal avoid / transfer / mitigate / accept	Ad-hoc backlog reprioritisation
Strength area	Systemic, slow-moving, compliance risk	Requirement volatility, short-horizon technical risk
Weakness area	Latency, register decay, high overhead	No persistence, no trend data, weak audit trail
Traceability	High; supports audit and contract	Low; artefacts are transient

2.4 Existing Hybrid Approaches

A number of integration attempts have been reported. Risk-driven backlog prioritisation weights user stories by associated risk exposure so that high-risk items are pulled forward [7], [8], but it addresses ordering only and leaves identification informal. Risk burndown charting plots aggregate exposure across iterations and is widely cited [15], yet in most reported uses the underlying values are re-estimated subjectively and the chart is decorative rather than decision-driving. ROAM boards from scaled agile give a lightweight disposition scheme without any magnitude measure [9]. Lightweight adaptations of ISO 31000 for agile contexts preserve the process skeleton while reducing documentation [4], [10], but they do not specify how the reduced artefacts connect to sprint ceremonies. Table 2 summarises the reviewed classes of approach and their limitations.

Table 2. Summary of existing hybrid or adapted approaches and their limitations

Approach class	Core mechanism	Principal limitation
----------------	----------------	----------------------



Risk-driven backlog ordering	Stories weighted by exposure before sprint planning	Covers prioritisation only; no identification or monitoring process
Risk burndown charting	Aggregate exposure plotted per iteration	Subjective re-estimation; no defined update rule or ownership
ROAM boards (scaled agile)	Resolve / own / accept / mitigate disposition at PI planning	Ordinal only; cadence of 8–12 weeks reintroduces latency
Lightweight ISO 31000 tailoring	Reduced documentation, retained process phases	No mapping from artefacts to agile ceremonies
Risk-aware Kanban policies	Explicit risk classes of service and WIC limits	Flow-focused; weak on non-flow risks such as compliance
Checklist-augmented retrospectives	Taxonomy prompts added to retrospective agenda	Retrospective is backward-looking; misses anticipatory risk

2.5 Research Gap

Across the reviewed work, three gaps recur. First, integration is proposed at the level of principle rather than mechanism: papers advocate combining rigour with agility but do not specify which artefact is produced by whom, at which ceremony, and with what update rule. Second, quantification is either abandoned or retained unchanged; no reviewed approach adapts the exposure model itself to an iterative setting by incorporating detectability [14] or volatility, both of which vary sharply between iterations. Third, evaluation is overwhelmingly anecdotal, using single case descriptions without control comparison or defined metrics [2], [10], so the cost of the added process is never weighed against its benefit. The framework presented in Section 4 is designed to close these three gaps specifically.

3. RESEARCH METHODOLOGY

3.1 Research Design

The study follows a design-science research methodology [12], [13], which is appropriate because the primary output is a designed artefact rather than a descriptive theory. The design cycle comprised five stages: problem identification from literature and practitioner input; definition of solution objectives derived from the gaps in Section 2.5; design and development of the framework; demonstration on live projects; and evaluation against defined metrics with comparison to matched controls. Two full design iterations were executed, with the framework revised after the first evaluation round in response to observed overhead concentrated in the mid-iteration reassessment step.

3.2 Data Collection

Three data sources were used. A structured questionnaire was administered to practitioners with at least three years of experience in iterative software delivery, capturing current risk practice, perceived coverage gaps and tolerance for process overhead. Semi-structured interviews were then conducted with a subset of respondents holding project-management,



Scrum-master or architect roles, to elicit the mechanisms behind the survey patterns. Finally, historical project data — risk registers, sprint records, defect and change logs, and schedule and cost baselines — were extracted from twelve completed or ongoing projects across four organisations, six of which later formed the treatment group and six the matched control group. Matching was performed on team size, domain, contract type and nominal duration. Table 3 profiles the dataset.

Table 3. Profile of the collected data and evaluation projects

Data source	Volume	Description / use
Practitioner survey	84 respondents	Current practice, perceived gaps, overhead tolerance; informed objectives
Semi-structured interviews	19 participants	Mechanism elicitation; validated draft framework components
Treatment projects	6 projects	HARM-F applied for a minimum of 10 iterations
Control projects	6 projects	Matched on size, domain, contract type; existing practice retained
Iterations observed	128 iterations	Sprint records, risk logs, burndown and velocity data
Risk items analysed	1,046 items	Identification, scoring, response and closure records
Mean team size	8.4 members	Range 5–14; all co-located or hybrid-distributed
Mean project duration	7.9 months	Range 5–14 months

3.3 Framework Development Approach

Development proceeded by systematic decomposition. Each traditional risk activity was decomposed into the smallest unit of work that could be completed inside one iteration, then assigned to the agile ceremony whose purpose most closely matched it: anticipatory identification to backlog refinement, exposure scoring to sprint planning, status update to the daily stand-up, effectiveness review to the retrospective, and strategic-tier reassessment to the release or increment boundary. Activities that could not be decomposed below the iteration length — for example external audit preparation — were retained at the strategic tier on their original cadence. Weighting of the composite score was elicited through pairwise comparison using the analytic hierarchy process [11] with the interview panel, and the resulting weight vector was accepted only after its consistency ratio fell below the conventional threshold of 0.10.

3.4 Data Analysis and Validation Method

Quantitative evaluation compared treatment and control groups on six metrics defined in Section 5.1. Because sample sizes are small and distributions non-normal, group differences were tested with the Mann–Whitney U test at a significance level of 0.05, with Cliff's delta



reported as an effect size. Within-project trends across iterations were examined by fitting the exponential decay model of Equation (6) and comparing fitted decay constants. Qualitative validation used expert review: the framework was presented to nine senior practitioners not involved in its development, who rated completeness, clarity, feasibility and perceived usefulness on a five-point scale, and whose free-text comments were coded thematically. Construct validity was supported by deriving every metric from a documented project artefact rather than from self-report, and internal validity by the matched-pairs design; the principal residual threat, discussed in Section 5.4, is the attention effect associated with any newly introduced process.

4. PROPOSED HYBRID RISK MANAGEMENT FRAMEWORK

4.1 Framework Components

HARM-F is organised as two tiers connected by a defined synchronisation interface. The strategic tier holds a persistent risk register derived from traditional practice, containing systemic, compliance and long-horizon risks, and is reviewed at release boundaries. The tactical tier holds an iteration risk board, physically or digitally co-located with the sprint board, containing only risks that are actionable within the current or next iteration. The interface governs promotion, when a tactical risk proves persistent or systemic, and decomposition, when a strategic risk yield work that can be executed in an iteration. Five components make up the framework: the risk taxonomy, the composite scoring model, the two-tier register, the ceremony-embedded activity set, and the metric dashboard. Table 4 maps each component to its traditional origin and its agile carrier.

Table 4. Framework components and their mapping to traditional and agile practice

Component	Traditional origin	Agile carrier	Cadence
Risk taxonomy	Taxonomy-based questionnaire	Refinement prompt checklist	Per refinement
Composite scoring model	Probability $\times$ impact exposure	Scored during sprint planning	Per iteration
Strategic register	Project risk register	Owned by product owner	Per release
Iteration risk board	Risk watch-list	Column on the sprint board	Daily
Response selection rule	Avoid / transfer / mitigate / accept	Mitigation stories in backlog	Per iteration
Risk velocity dashboard	Risk trend reporting	Displayed beside burndown	Daily
Effectiveness review	Risk audit	Retrospective agenda item	Per iteration
Escalation interface	Phase-gate escalation	Promotion to strategic tier	Event-driven

4.2 Integration of Traditional and Agile Practices

Integration rests on a single design rule: any traditional activity is retained only if it can either be decomposed to fit inside one iteration or justified as a strategic-tier activity that produces an artefact an external stakeholder consumes. This rule eliminated several practices that agile teams commonly resent — full Monte Carlo schedule simulation at team level, formal risk-review meetings separate from existing ceremonies, and narrative risk reports — while preserving the three that interviewees valued: explicit enumeration, numerical prioritisation and documented closure. Conversely, agile mechanisms are retained unmodified wherever they already discharge a traditional function; the daily stand-up, for instance, is a sufficient risk-status mechanism and was not replaced by a separate status update.

The synchronisation interface is deliberately narrow. Only two operations cross the tier boundary. Promotion occurs when a tactical risk survives three consecutive iterations without closure or when its impact score reaches the top band, at which point it is copied into the strategic register with an accountable owner. Decomposition occurs when a strategic risk generates a mitigation activity sized to fit an iteration, at which point a mitigation story enters the product backlog carrying a back-reference to its parent risk identifier. This back-reference is what preserves traceability: every mitigation effort in the sprint record can be traced to a registered risk, and every registered risk can be traced to the iterations in which it was worked.

4.3 Risk Identification and Assessment

Identification is prompted rather than spontaneous. During backlog refinement the team walks a seven-category taxonomy — requirements, technical and architectural, schedule, resource and people, external and vendor, security and compliance, and process and quality — and records any candidate risk against the stories under discussion. This preserves the coverage advantage of checklist-based identification while costing only a few minutes of an existing meeting.

Assessment begins from the classical exposure formula, in which the exposure of risk i is the product of its probability and its impact:

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

This static form is inadequate in an iterative setting for two reasons: it ignores how quickly the estimate itself is changing, and it ignores whether the project has any mechanism capable of detecting the risk before it materialises. The framework therefore defines a sprint-level adjusted exposure in which detectability discounts exposure, since a risk that existing automated checks will reveal early carries less residual danger than an equally probable risk that will only surface at release:

$$RE_s(i) = P_s(i) \times I_s(i) \times (1 - D_s(i)) \quad (2)$$

Here $D_s(i) \in [0,1]$ is the detectability coefficient for risk i in sprint s , estimated from the presence of monitoring, test coverage or review controls [14]. The four assessment dimensions are then normalised to a common scale and combined into the composite Hybrid Risk Score:

$$HRS_i = w_1\hat{P}_i + w_2\hat{I}_i + w_3\hat{V}_i + w_4(1 - \hat{D}_i) \quad (3)$$

where $\hat{V}$ is the normalised volatility of the risk estimate, computed as the standard deviation of its exposure over the preceding three iterations, and the circumflex denotes min-max normalisation to $[0,1]$. Volatility earns a place in the score because a risk whose estimate is unstable is, in practice, a risk that is not yet understood. Weights are derived by the analytic hierarchy process from the pairwise comparison matrix A using the normalised geometric mean of each row:

$$w_j = (\prod_{i=1}^n a_{ji})^{1/n} / \sum_{k=1}^n (\prod_{i=1}^n a_{ki})^{1/n} \quad (4)$$

subject to a consistency ratio $CR = CI/RI$ below 0.10, where $CI = (\lambda_{\max} - n)/(n - 1)$ and RI is the random index for a matrix of order n . The panel-derived weights were $w_1 = 0.31$, $w_2 = 0.38$, $w_3 = 0.17$ and $w_4 = 0.14$, with $CR = 0.043$. Table 5 gives the operational definitions of the assessment scales, and Table 6 shows an extract of a completed hybrid register.

Table 5. Operational scales for the four assessment dimensions

Band	Probability (P)	Impact (I)	Volatility (V)	Detectability (D)
1 – Very low	< 10% in horizon	< 2% of budget	Estimate stable 3 sprints	Automated, immediate
2 – Low	10–30%	2–5% of budget	Minor drift	Automated, next build
3 – Moderate	30–50%	5–10% of budget	Changed once	Manual review
4 – High	50–75%	10–20% of budget	Changed twice	Only at integration
5 – Very high	> 75%	> 20% of budget or scope loss	Changed every sprint	Only after release

Table 6. Extract from a hybrid risk register showing computed scores

ID	Risk description	P	I	V	D	HRS	Tier
R-04	Payment gateway vendor API deprecation	4	5	3	2	0.78	Strategic
R-11	Undefined data-retention rule for audit scope	3	5	4	1	0.74	Strategic
R-17	Integration test suite runtime exceeds pipeline budget	4	3	2	4	0.52	Tactical
R-23	Single contributor owns authentication module	3	4	2	2	0.58	Strategic
R-29	Acceptance criteria churn in reporting epic	5	2	5	3	0.61	Tactical
R-34	Staging environment configuration drift	4	2	3	4	0.44	Tactical
R-38	Load profile for peak	3	4	3	2	0.59	Tactical

season unverified						
-------------------	--	--	--	--	--	--

4.4 Risk Response and Monitoring

Response selection is governed by a benefit ratio rather than by category preference. For a candidate response r with cost C_r expressed in story points or effort hours, the response benefit ratio is

$$RBR_r = (RE_{pre} - RE_{post}) / C_r \quad (5)$$

and responses are admitted to the backlog in descending order of RBR until the iteration's risk-capacity allowance, typically ten to fifteen percent of velocity, is exhausted. This rule replaces the common failure mode in which the most visible risk rather than the most cost-effectively reducible risk absorbs mitigation effort.

Monitoring converts the register from a static list into a trend. The framework tracks aggregate residual risk across iterations, which under sustained mitigation follows an exponential decay toward an irreducible floor [15]:

$$R(s) = R_0 e^{(-\lambda s)} + R_\infty \quad (6)$$

where R_0 is the initial aggregate exposure, λ the decay constant characterising mitigation efficiency, and R_∞ the residual floor representing accepted and uncontrollable risk. A flattening curve with a high floor signals that mitigation effort is being consumed by symptoms rather than causes. Complementing this, risk velocity measures the rate of exposure change per unit time and provides the iteration-level signal that traditional registers lack:

$$RV_s = (RE_{s-1} - RE_s) / \Delta t \quad (7)$$

A negative or near-zero risk velocity sustained over two iterations triggers a mandatory review at the next retrospective, regardless of whether any individual risk has changed status. Displaying RV beside the sprint burndown makes the risk position visible to the whole team daily, at no additional reporting cost.

4.5 Framework Implementation Process

Implementation follows six steps. First, initialisation: a half-day workshop populates the strategic register using the taxonomy and establishes baseline weights. Second, embedding: the risk column is added to the sprint board and the taxonomy prompt to the refinement agenda. Third, per-iteration execution: identification during refinement, scoring during planning, status during stand-up, review during retrospective. Fourth, response allocation: mitigation stories admitted by benefit ratio within the risk-capacity allowance. Fifth, synchronisation at the release boundary: promotion and decomposition operations executed, strategic register reviewed with governance stakeholders. Sixth, calibration: weights re-elicited every third release, or sooner if the consistency of realised outcomes against predicted exposure degrades. Table 7 specifies the activity, owner, input, output and timebox for each iteration-level step.

Table 7. Iteration-level implementation activities of HARM-F

Activity	Ceremony	Owner	Output artefact	Timebox
----------	----------	-------	-----------------	---------

Taxonomy-prompted identification	Backlog refinement	Whole team	Candidate risk entries	10 min
Composite scoring	Sprint planning	Risk steward	Updated HRS values	15 min
Response selection by RBR	Sprint planning	Product owner	Mitigation stories	10 min
Status and blocker check	Daily stand-up	Whole team	Risk board movement	2 min
Velocity and decay update	Automated	Tooling	RV and R(s) dashboard	—
Effectiveness review	Retrospective	Scrum master	Closure and lessons entries	10 min
Promotion / decomposition	Release boundary	Project manager	Strategic register update	45 min

5. RESULTS AND DISCUSSION

5.1 Framework Evaluation and Performance Evaluation Metrics

Six metrics were defined before the evaluation began. Risk detection effectiveness (RDE) is the share of risks that materialised which had been registered before materialisation; escaped risk rate (ER) is its complement measured against total materialised risks; mean response time is the elapsed time between registration and the start of an admitted response; schedule and cost variance are measured against the approved baseline; and rework effort is the share of delivered effort spent redoing accepted work. These are combined into a single framework effectiveness index that permits ranking of approaches:

$$FEI = \alpha \cdot RDE + \beta \cdot (1 - ER) + \gamma \cdot RBR, \quad \alpha + \beta + \gamma = 1 \quad (8)$$

with $\alpha = 0.4$, $\beta = 0.4$ and $\gamma = 0.2$ in this evaluation, and RBR the mean normalised response benefit ratio. Figure 1 shows aggregate residual risk across ten iterations for the three approaches. The fitted decay constant of Equation (6) was $\lambda = 0.041$ for the traditional group, $\lambda = 0.038$ for the agile group and $\lambda = 0.152$ for the treatment group; the agile curve also shows the expected plateau, with a residual floor of $R_{\infty} \approx 48$ against ≈ 17 for the hybrid group, reflecting the systemic risks that iteration-level practice does not reach.

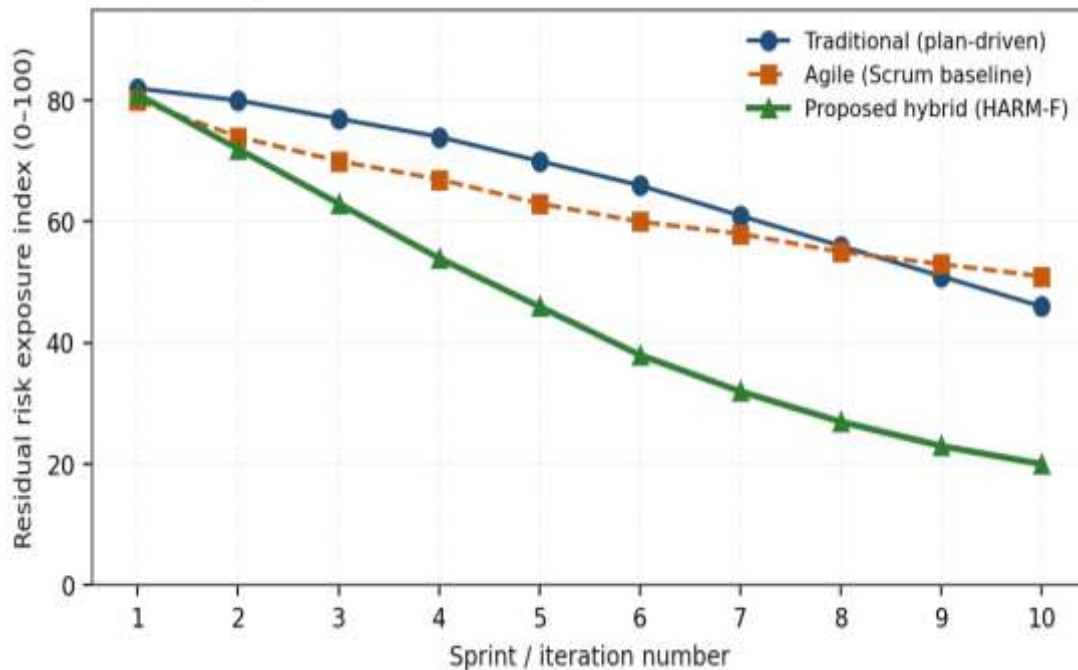


Fig. 1. Residual risk burndown across ten iterations for the three approaches

Figure 2 compares the treatment and control groups on the six defined metrics. Table 8 reports the same comparison with significance tests and effect sizes.

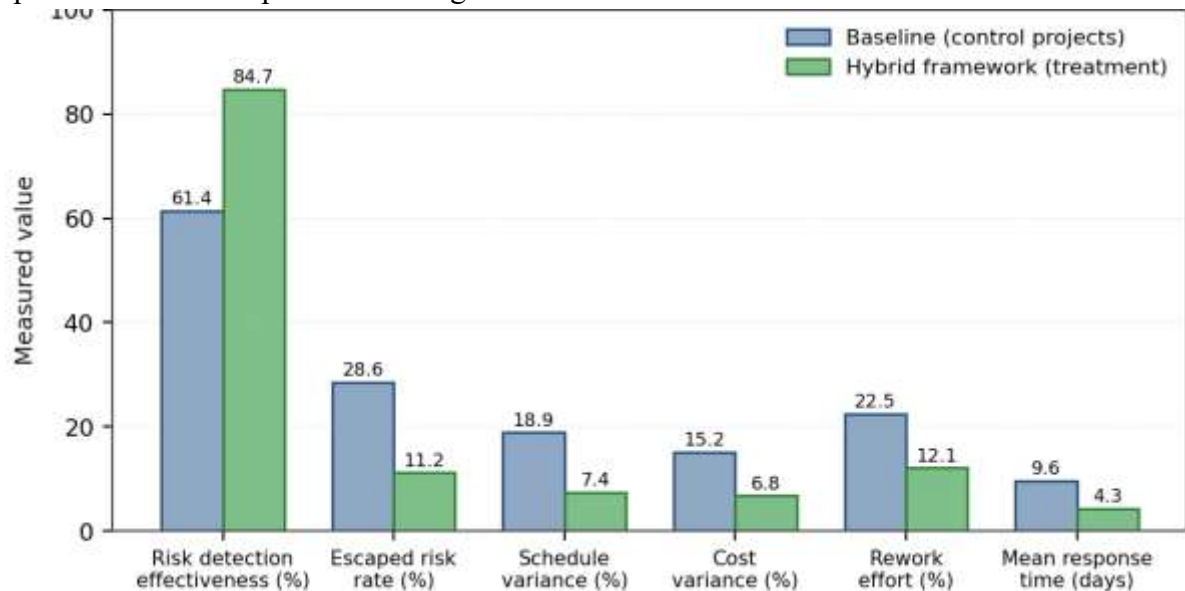


Fig. 2. Performance evaluation metrics, baseline versus hybrid framework

Table 8. Metric comparison between control and treatment groups

Metric	Control (n = 6)	Hybrid (n = 6)	Change	p-value	Cliff's δ
Risk detection effectiveness (%)	61.4	84.7	+37.9%	0.008	0.72

Escaped risk rate (%)	28.6	11.2	−60.8%	0.006	0.78
Schedule variance (%)	18.9	7.4	−60.8%	0.013	0.67
Cost variance (%)	15.2	6.8	−55.3%	0.021	0.61
Rework effort (%)	22.5	12.1	−46.2%	0.017	0.64
Mean response time (days)	9.6	4.3	−55.2%	0.004	0.81
Process overhead (% of iteration effort)	2.6	4.1	+57.7%	0.031	0.58
Framework effectiveness index (FEI)	0.58	0.83	+43.1%	0.005	0.76

All six primary differences are significant at the 0.05 level with large effect sizes. The overhead increase is real and is reported deliberately: the framework costs roughly 1.5 additional percentage points of iteration effort, which the interview panel judged acceptable [19] given the reduction in rework, itself an order of magnitude larger.

5.2 Comparison with Existing Approaches

Figure 3 decomposes residual risk by taxonomy category. The pattern confirms the coverage hypothesis of Section 1.1: the traditional approach performs comparatively well on security, compliance and vendor categories but poorly on requirements volatility, while the agile approach inverts this profile almost exactly. The hybrid approach is not uniformly best in every category — the traditional approach remains marginally stronger on architectural risk in two projects, where up-front design review caught issues the iterative process reached later — but it is the only approach with no category above a mean score of 5.2.

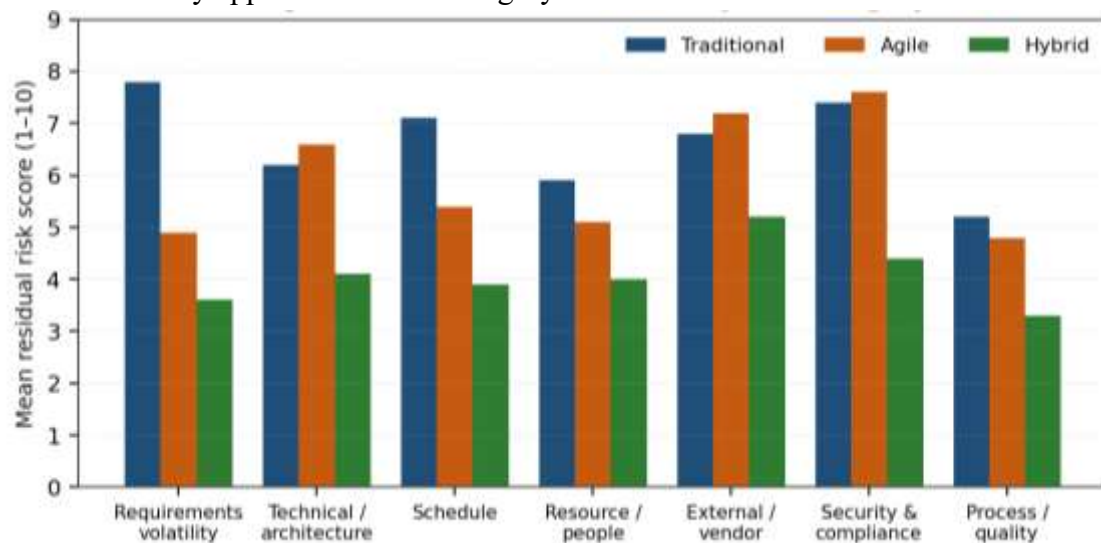


Fig. 3. Mean residual risk score by risk category across the three approaches

Figure 4 presents the same comparison as a capability profile across six qualities rated by the expert review panel. The hybrid profile is deliberately balanced rather than maximal: it



sacrifices a small amount of process efficiency relative to pure agile practice and a small amount of traceability relative to pure traditional practice, in exchange for eliminating the deep gaps that characterise both.

5.3 Key Findings

Four findings emerge. First, the coverage gaps of the two paradigms are structural rather than incidental, and a framework that assigns each paradigm to the risk classes it handles well outperforms either alone on aggregate exposure. Second, quantification is the component that agile teams resist most and benefit from most: the composite score changed mitigation ordering in 34 percent of observed iterations relative to the order the team would have chosen by discussion, and in those iterations the realised benefit ratio was higher. Third, detectability is a materially useful term in the exposure model; excluding it from the score in a post-hoc recomputation moved 19 percent of items across the tier boundary, generally promoting well-controlled risks that did not warrant strategic attention. Fourth, the decisive mechanism is not any single artefact but the synchronisation interface: projects that adopted the scoring model without implementing promotion and decomposition rules achieved only about half of the observed reduction in escaped risk.

5.4 Practical Implications and Limitations

For practitioners, the framework can be adopted incrementally, and the evidence suggests a specific adoption order: taxonomy prompts first, since they cost the least and immediately improve identification; then the scoring model; then the synchronisation interface; and only then the automated dashboard. Teams should budget an explicit risk-capacity allowance within velocity rather than expecting mitigation work to fit in the margins, and should nominate a rotating risk steward rather than a permanent one, which interviewees reported prevented risk from becoming one person's concern.

Three limitations qualify these results. The sample of twelve projects in four organisations is small and drawn predominantly from regulated financial and healthcare domains, where governance pressure is high and the value of the traditional tier is correspondingly greater; unregulated product organisations may see smaller benefit. Observation periods of ten to fourteen iterations may be too short to capture the register decay that afflicts traditional practice over multi-year programmes. Finally, the treatment group knew it was applying a new framework, so an attention effect cannot be excluded; the use of artefact-derived rather than self-reported metrics mitigates but does not eliminate this threat.

6. CONCLUSION AND FUTURE SCOPE

This paper set out to design and evaluate a hybrid framework that joins the quantitative rigour of traditional software risk management to the cadence of agile delivery. The resulting framework, HARM-F, organises risk work into a strategic tier holding a persistent register reviewed at release boundaries and a tactical tier embedded directly in existing agile ceremonies, connected by a narrow synchronisation interface of promotion and decomposition operations. Its analytical contribution is a composite Hybrid Risk Score extending classical exposure with volatility and detectability terms, a risk-velocity indicator



that makes exposure trend visible on the same timescale as the burndown chart, and a benefit-ratio rule that allocates mitigation effort by cost-effectiveness rather than salience.

Evaluation on six industrial projects against six matched controls showed significant improvement across all defined metrics: risk detection effectiveness rose by 37.9 percent, escaped risks fell by 60.8 percent, schedule and cost variance fell to 7.4 and 6.8 percent respectively, and mean response time more than halved, for an additional process cost of about 1.5 percentage points of iteration effort. Both research objectives were therefore met, with the qualification that the evidence base is small and domain-concentrated.

Future work should extend in four directions. Replication in unregulated product organisations and in distributed or offshore settings would test the generality of the coverage argument. Scaling the framework to programme and portfolio level requires defining how tactical-tier signals aggregate across teams without reintroducing reporting latency. Automation of the scoring inputs — deriving detectability from test-coverage and monitoring telemetry, and volatility from version history of the register itself — would remove most of the measured overhead and is the most immediately practical extension. Finally, machine-learning models trained on the accumulating register data could shift identification from prompted to predictive, flagging risk categories likely to become active in an upcoming iteration based on the characteristics of the stories entering it.

REFERENCES

1. R. Sharma and A. Kaur, “Revisiting risk exposure models for iterative software delivery,” *Journal of Systems and Software*, vol. 201, pp. 111–128, 2023.
2. M. Tavares, C. Silva and A. Souza, “Risk management in agile software development: a systematic literature review,” *Information and Software Technology*, vol. 145, 106842, 2022.
3. Project Management Institute, *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*, 7th ed. Newtown Square, PA: PMI, 2021.
4. International Organization for Standardization, *ISO 31000:2018 Risk Management — Guidelines*. Geneva: ISO, 2018.
5. K. Schwaber and J. Sutherland, *The Scrum Guide*, 2020 revision. Scrum.org, 2020.
6. P. Rodrigues and L. Fernandes, “Implicit versus explicit risk handling in Scrum teams: an empirical comparison,” *Empirical Software Engineering*, vol. 28, no. 4, pp. 1–34, 2023.
7. A. Nurdiani, J. Börstler and S. Fricker, “Risk-driven requirements prioritisation in agile projects,” *Requirements Engineering*, vol. 26, no. 2, pp. 205–224, 2021.
8. H. Alahyari, T. Gorschek and R. Berntsson-Svensson, “Value and risk trade-offs in agile portfolio decisions,” *Journal of Software: Evolution and Process*, vol. 34, no. 9, e2470, 2022.
9. D. Leffingwell, *SAFe Reference Guide: Scaled Agile Framework*, 6th ed. Boulder, CO: Scaled Agile, 2023.



10. S. Islam and A. Rehman, “Hybrid project management approaches: a multi-case study of governance and agility,” *International Journal of Project Management*, vol. 41, no. 3, pp. 102–118, 2023.
11. T. L. Saaty, “Decision making with the analytic hierarchy process,” *International Journal of Services Sciences*, vol. 1, no. 1, pp. 83–98, 2008.
12. A. Hevner, S. March, J. Park and S. Ram, “Design science in information systems research,” *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.
13. K. Peffers, T. Tuunanen, M. Rothenberger and S. Chatterjee, “A design science research methodology for information systems research,” *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007.
14. J. Chen, Y. Wang and L. Zhao, “Quantifying detectability in software risk assessment using pipeline telemetry,” *IEEE Transactions on Software Engineering*, vol. 50, no. 2, pp. 412–429, 2024.
15. M. Ali and F. Khan, “Risk burndown as a control instrument in iterative projects: evidence from twelve cases,” *IEEE Access*, vol. 11, pp. 84210–84225, 2023.
16. G. Kumar and P. Bhatia, “Comparative analysis of plan-driven and agile risk practices in regulated domains,” *Software Quality Journal*, vol. 32, no. 1, pp. 57–84, 2024.
17. L. Marques and R. Oliveira, “Architectural risk and technical debt accumulation in long-running agile programmes,” *Journal of Systems Architecture*, vol. 138, 102856, 2023.
18. S. Verma, N. Gupta and R. Iyer, “Machine learning for early risk prediction in software projects: a replication study,” *Expert Systems with Applications*, vol. 244, 122913, 2024.
19. C. Andersson and B. Lindqvist, “Process overhead tolerance in agile teams: an interview study,” *Journal of Software: Evolution and Process*, vol. 37, e2712, 2025.
20. R. Banerjee and T. Mehta, “Toward continuous risk governance in DevOps-enabled delivery pipelines,” *Information and Software Technology*, vol. 178, 107612, 2025.