



SP-CARE: A Risk-Aware Framework for Continuous System Prompt Evaluation and Adaptive Revalidation in Agentic AI Systems

Dr. Sanjay Nakharu Prasad Kumar

The George Washington University, USA

Publication History: Received: 25.08.2026; Revised: 24.09.2026; Accepted: 29.09.2026; Published: 08.10.2026.

ABSTRACT: System prompts have become a consequential behavioral control layer in large language model applications and agentic AI systems. They shape instruction adherence, task completion, tool selection, verification, safety boundaries, escalation, style, and resource use. Yet system-prompt changes are commonly assessed with a small number of manually selected examples, static benchmarks, or uncalibrated model-based judgments. These practices provide limited visibility into rare but severe failures, interactions among instructions, prompt-model coupling, non-deterministic behavior, tool-mediated risk, and regressions introduced by model or harness upgrades. This paper proposes SP-CARE, a System Prompt Continuous Assessment and Risk Evaluation framework that treats the system prompt as a versioned production control artifact. SP-CARE combines requirement decomposition, risk-based test portfolio construction, controlled repeated experiments, multi-source evaluation, release gating, production monitoring, and adaptive revalidation. It defines a multidimensional taxonomy covering instruction fidelity, task effectiveness, claim reliability, tool behavior, verification discipline, safety, recovery, robustness, user experience, and operational efficiency. The framework also introduces a risk-adjusted decision index that penalizes execution variability, tail-failure severity, cost, and latency while enforcing non-compensatory safety gates. A reproducible empirical protocol is specified for comparing SP-CARE with static benchmark and single-judge baselines across prompt variants, model configurations, tool environments, and repeated runs. The intended contribution is not a claim of completed empirical superiority, but a falsifiable framework and research design for determining whether a system-prompt change is reliable enough to release and portable enough to survive system evolution.

KEYWORDS: system prompt evaluation; large language models; agentic AI; LLM-as-a-judge; prompt robustness; behavioral testing; AI governance; model regression; tool-use evaluation; risk-aware evaluation

I. INTRODUCTION

Large language model (LLM) systems are increasingly configured through persistent instructions that define roles, constraints, priorities, tool policies, output requirements, and safety boundaries. In a simple conversational application, these instructions shape tone and response structure. In an agentic application, the same layer may influence whether a model reads evidence before making a claim, selects an appropriate tool, requests confirmation before a destructive action, verifies a change, recognizes repeated failure, or stops an unsafe workflow. A system prompt is therefore more than a writing aid. It is part of the executable behavioral specification of an AI-enabled product.

System-prompt behavior is difficult to validate exhaustively because it emerges from interactions among the prompt, model, agent harness, tools, context, task, user behavior, and stochastic decoding. An apparently local wording change can improve one scenario while degrading another. A model upgrade can expose instructions that were previously benign, follow ambiguous instructions more literally, or change how conflicts are resolved. Tools and retrieved content add additional paths through which the same prompt can produce different outcomes. Recent production experience reported by the Kiro team illustrates this problem: benchmark signals were supplemented with analysis of internal conversations, controlled prompt cohorts, behavioral rubrics, and revalidation after a model upgrade [1].

Existing research provides important components of a solution but not a complete system-prompt lifecycle. HELM demonstrates the need for broad scenarios and multiple metrics rather than a single accuracy value [2]. CheckList shows how behavioral testing can expose failures missed by held-out accuracy [3]. Prompt-format and adversarial-prompt studies demonstrate that semantically similar instructions can yield materially different behavior [4], [5]. Prompt Risk Control provides statistically principled methods for selecting prompts under chosen risk bounds [6].



LLM-as-a-judge methods scale evaluation of open-ended outputs, but they introduce position, verbosity, self-preference, and calibration concerns [7]-[10]. Agent benchmarks expand evaluation from text outputs to interactive trajectories [11], [12], while security research shows that tool-using systems can be compromised through prompt injection and untrusted external content [13]-[15].

The unresolved challenge is operational and scientific: how should an organization determine whether a new system-prompt version is better, safe enough to release, stable across repeated executions, and transferable across changing models and tools? Static benchmark scores are insufficient because they can hide tail failures, average away instability, omit tool behavior, or fail to represent production conditions. Production feedback alone is also insufficient because it is observational, delayed, selectively expressed, and potentially confounded. A rigorous approach must connect the two.

This paper proposes SP-CARE, a System Prompt Continuous Assessment and Risk Evaluation framework. SP-CARE is designed as a lifecycle rather than a one-time score. It converts prompt language into traceable behavioral requirements; constructs representative, boundary, metamorphic, adversarial, regression, and longitudinal tests; executes controlled and repeated prompt comparisons; combines deterministic validators, task metrics, calibrated LLM judges, human adjudication, and user signals; and establishes release gates and revalidation triggers. The framework explicitly treats prompt performance as conditional on the model-harness configuration and introduces a risk-adjusted index that penalizes variability and tail loss rather than rewarding only mean performance.

1.1 Research Objective

The objective is to develop a falsifiable, production-oriented framework for evaluating human-authored system prompts in agentic AI systems. The framework must answer three decision questions:

1. Does the candidate prompt improve the intended behaviors relative to a stable control?
2. Does it introduce unacceptable regressions, tail risks, or operational costs?
3. Do the observed improvements remain valid when the model, tools, context, or user interaction pattern changes?

1.2 Research Gap

Five gaps motivate SP-CARE. First, prompt evaluation often lacks requirement-level traceability, making it difficult to identify which instruction caused a behavioral failure. Second, benchmark coverage is rarely organized around a formal risk and behavior taxonomy. Third, open-ended outcomes are frequently judged through a single evaluator without sufficient human calibration or bias testing. Fourth, most aggregate scores underrepresent execution variability and severe low-frequency events. Fifth, prompt changes are not consistently revalidated as the surrounding system evolves.

1.3 Contributions

This paper makes six proposed contributions:

1. A system-prompt-specific lifecycle that joins offline testing, controlled experimentation, production signals, and adaptive revalidation.
2. A requirement decomposition method that maps prompt clauses to intended behavior, evidence, severity, and test coverage.
3. A portfolio model that organizes representative, boundary, metamorphic, adversarial, regression, and longitudinal tests.
4. A multidimensional evaluation taxonomy spanning outcomes, trajectories, safety, robustness, user experience, and operations.
5. A risk-adjusted System Prompt Assurance Index that includes mean performance, instability, conditional tail loss, cost, latency, and hard safety gates.
6. A reproducible empirical protocol and statistical analysis plan for evaluating whether SP-CARE detects more consequential regressions and aligns more closely with expert judgment than simpler baselines.

II. BACKGROUND AND RELATED WORK

2.1 Holistic and Behavioral Evaluation

LLM evaluation has moved from narrow task accuracy toward broader scenario and metric coverage. HELM formalizes holistic evaluation through scenario taxonomies, standardized conditions, and multiple desiderata including accuracy, robustness, fairness, toxicity, calibration, and efficiency [2]. Its core lesson is methodological: evaluation should expose trade-offs and missing coverage rather than compressing diverse system properties into an unexplained headline number.



CheckList brings concepts from software testing into NLP through minimum functionality tests, invariance tests, and directional expectation tests [3]. The approach is relevant to system prompts because prompt instructions frequently imply behavioral invariants. For example, a harmless change in user phrasing should not change whether the agent asks before deleting a file, and the addition of irrelevant context should not reverse a grounded answer. CheckList also demonstrates that structured test ideation can reveal actionable defects even in extensively tested systems.

SP-CARE extends these ideas from model-level behavior to prompt-version decisions. It assigns every high-risk prompt requirement to one or more tests and distinguishes coverage of a behavior from the frequency with which that behavior appears in a convenience sample.

2.2 Prompt Sensitivity, Robustness, and Risk Control

Prompt sensitivity challenges the assumption that a fixed prompt is a neutral evaluation interface. Sclar et al. report substantial performance variation across meaning-preserving formatting choices and show that prompt-format effects transfer weakly across models [4]. PromptRobust evaluates character-, word-, sentence-, and semantic-level perturbations and finds that small changes can materially affect performance [5]. These findings support two SP-CARE design choices: metamorphic testing of meaning-preserving prompt and user-input variants, and explicit measurement of prompt-model interaction.

Prompt Risk Control (PRC) addresses a related but distinct problem: selecting prompts using statistically rigorous upper bounds for user-chosen risk measures, including mean, quantile, and disparity-oriented measures [6]. PRC is an important foundation for risk-aware prompt selection. SP-CARE differs in scope. It focuses specifically on versioned system prompts in agentic systems and adds requirement traceability, trajectory evaluation, tool and permission behavior, production-derived failure loops, release governance, and model-upgrade revalidation. SP-CARE's proposed index is a decision aid rather than a substitute for formal distribution-free risk bounds. Where sufficient labeled data exist, PRC-style bounds can be embedded as an evaluator within SP-CARE.

2.3 LLM-as-a-Judge and Calibration

Open-ended system behavior cannot be fully evaluated through exact match. Zheng et al. show that capable LLM judges can approximate human preferences at useful scale, while documenting position, verbosity, self-enhancement, and reasoning biases [7]. G-Eval uses structured criteria and a form-filling approach to improve alignment with human ratings but also observes possible bias toward model-generated text [8]. LLM-Rubric demonstrates that multidimensional rubric outputs can be calibrated to human judges and that uncalibrated model judgments may not adequately represent human standards [9]. Position-bias research further shows that repetition stability and order swapping are necessary when assessing pairwise judges [10].

These findings argue against treating an LLM judge as ground truth. SP-CARE uses judges as one evidence source, requires atomic criteria, hides treatment identity, swaps pairwise ordering, repeats low-confidence judgments, and calibrates against a human-labeled sample. Critical safety and authorization decisions are not delegated solely to a generative judge.

2.4 Agent and Tool-Use Evaluation

AgentBench expands evaluation into interactive environments and identifies long-horizon reasoning, decision-making, and instruction following as common obstacles [11]. Tau-bench evaluates tool-agent-user interactions under domain policies and introduces pass^k to measure the probability that an agent succeeds consistently across repeated trials [12]. These benchmarks show why a response-only evaluation is insufficient for agentic systems. A plausible final answer may conceal an unauthorized action, an incorrect tool parameter, unnecessary data access, or a failed intermediate step.

SP-CARE therefore treats the trajectory as a first-class evaluation object. The evaluator records tool selection, arguments, observations, retries, side effects, confirmations, terminal state, and the evidence supporting the final claim. Outcome and process are scored separately.

2.5 Security, Trustworthiness, and Governance

Tool-using agents process data that may contain instructions. Greshake et al. show that indirect prompt injection can exploit the blurred boundary between data and commands in LLM-integrated applications [14]. AgentDojo provides a dynamic environment for measuring utility and security under prompt injection attacks [13]. DecodingTrust demonstrates that stronger task performance does not imply uniform trustworthiness and that jailbreaks, privacy leakage, adversarial robustness, and fairness require dedicated evaluation [15].



The NIST Generative AI Profile recommends lifecycle-oriented governance, measurement, pre-deployment testing, incident handling, and ongoing risk management [16]. SP-CARE translates these general principles into system-prompt controls: explicit risk ownership, non-compensatory gates, prompt and rubric versioning, audit evidence, post-release monitoring, and revalidation triggers.

2.6 Production Evaluation and the Remaining Gap

The Kiro case demonstrates an operational cycle that diagnoses recurring complaint categories, designs targeted prompt changes, tests isolated and combined cohorts, evaluates them with a consistent judge, and revalidates changes after model upgrades [1]. Its reported results are experiment-specific and appropriately presented as directional rather than universal. The case motivates continuous evaluation but leaves open several research questions: how to formalize prompt requirements, measure coverage, calibrate judges, quantify tail risk, compare model transfer, and decide when an apparent improvement is strong enough to release.

SP-CARE is intended to bridge that gap. It combines the behavioral specificity of CheckList, the breadth of HELM, the statistical concerns of PRC, the scalable judgment of LLM evaluators, the trajectory focus of agent benchmarks, and the lifecycle governance of NIST into a system-prompt-specific evaluation protocol.

III. PROBLEM FORMULATION

3.1 Definitions

A **system prompt** is the persistent natural-language instruction set supplied at a high-authority layer to shape an AI system's behavior across one or more interactions. The definition includes role, objectives, constraints, tool policies, safety conditions, output rules, escalation logic, and verification expectations. Product-specific developer instructions may be included when they serve the same persistent control function.

A **prompt version** is an immutable text and metadata artifact identified by a version identifier, hash, author, rationale, creation time, intended change, known risks, and compatibility range.

A **configuration** is the executable combination of prompt version, model, model parameters, agent harness, tool catalog, policy layer, context-construction logic, and environment state.

A **trajectory** is the ordered sequence of user messages, model outputs, tool calls, tool observations, state transitions, side effects, and terminal outcomes produced during a task.

A **behavioral requirement** is an atomic, testable statement derived from the system prompt. It defines the intended behavior, prohibited behavior, applicable conditions, expected evidence, severity, and pass/fail rule.

3.2 Conditional Nature of Prompt Performance

Let P denote a prompt version, M a model and inference configuration, H an agent harness, T a task, C the constructed context, U a user-interaction pattern, E the tool environment, and R a stochastic execution. The observed outcome vector Y is:

$$Y = f(P, M, H, T, C, U, E, R). \quad (1)$$

This formulation has two implications. First, a prompt cannot be described as universally good without specifying the configuration and task distribution. Second, isolating a prompt effect requires holding other components constant or modeling their interactions. SP-CARE therefore treats the prompt-model-harness configuration as the operational unit while using paired experiments to estimate the incremental effect of a prompt change.

3.3 Multi-Objective Quality

For evaluation dimension d in D , let s_d be a normalized score in $[0,1]$. Prompt evaluation is a multi-objective problem because a change can increase task completion while decreasing safety, consistency, or efficiency. The unadjusted quality vector is:

$$Q(P) = [s_1(P), s_2(P), \dots, s_D(P)]. \quad (2)$$

A scalar score may support release decisions, but it must not allow a high style or speed score to compensate for a critical authorization failure. SP-CARE consequently separates hard gates from compensatory quality metrics and reports both the vector and the risk-adjusted index.



3.4 Scope

SP-CARE evaluates human-authored system-prompt and configuration changes. It does not assume access to model weights and is compatible with API-hosted or locally served models. Automated prompt generation may propose candidates, but candidate generation is outside the core framework. The primary scope is interactive and tool-using applications; single-turn text generation is a supported special case.

IV. DESIGN REQUIREMENTS

The literature and problem formulation yield seven design requirements.

DR1 - Requirement traceability. Every high-impact prompt instruction must be represented as one or more atomic requirements, and every requirement must map to tests and evaluators.

DR2 - Risk-based coverage. Test selection must represent frequency, business impact, safety severity, uncertainty, and known production failures rather than convenience alone.

DR3 - Controlled isolation. Candidate prompts must be compared with stable controls using paired inputs, fixed environments, recorded configurations, and repeated runs.

DR4 - Evidence pluralism. Deterministic validators, task metrics, trajectory inspection, LLM judges, human adjudication, and user signals must be combined according to fitness for purpose.

DR5 - Tail-risk awareness. Evaluation must quantify variability and severe low-frequency failures, not only average performance.

DR6 - Lifecycle revalidation. A model, tool, policy, data, or harness change must be treated as a potential change to prompt behavior and must trigger proportionate retesting.

DR7 - Reproducibility and governance. Prompts, rubrics, test cases, judge configurations, tool schemas, outputs, and release decisions must be versioned and auditable.

V. THE SP-CARE FRAMEWORK

SP-CARE operationalizes the design requirements through five lifecycle phases: Specify, Portfolio, Execute, Assess, and Release/Revalidate. The name SP-CARE denotes System Prompt Continuous Assessment and Risk Evaluation; it is not limited to the initials of the five phases.

5.1 Phase 1: Specify and Decompose

The first phase transforms a natural-language prompt into a System Prompt Requirement Register. Long prompts often mix roles, priorities, constraints, examples, exceptions, tool rules, and style preferences. Treating the prompt as one undifferentiated artifact makes root-cause analysis difficult. Decomposition creates a testable representation without changing the prompt itself.

Each requirement record contains:

1. Requirement identifier and source clause.
2. Intended behavior and prohibited behavior.
3. Triggering conditions and exclusions.
4. Priority and conflict-resolution rule.
5. Severity and risk owner.
6. Observable evidence in output or trajectory.
7. Deterministic, model-based, or human evaluation method.
8. Associated test cases and release gate.

Requirements should be atomic enough that a single verdict is meaningful. For example, "work safely and completely" should be decomposed into separate requirements for reading relevant evidence, completing requested actions, verifying results, requesting authorization for destructive actions, and accurately reporting uncertainty. The register can also reveal contradictions, such as "proceed autonomously" and "ask before every tool call," that would otherwise create unstable behavior.

An instruction graph $G = (V, A)$ represents requirements V and relations A such as precedes, overrides, depends on, conflicts with, and applies only if. Graph analysis is not required for a minimal implementation, but it supports review of priority inversions and interaction effects.



5.2 Phase 2: Produce the Evaluation Portfolio

The evaluation portfolio is a stratified collection of tasks, contexts, environments, and expected behaviors. It contains six test families.

Representative tests sample common production tasks across user groups, domains, tool pathways, and complexity levels. They estimate ordinary effectiveness but should not dominate risk decisions merely because they are frequent.

Boundary tests target ambiguity, missing information, conflicting instructions, unusually long context, tool unavailability, partial permissions, and near-threshold conditions.

Metamorphic tests apply meaning-preserving changes to user wording, formatting, irrelevant details, message order, or context layout. Expected invariants and directional expectations are documented before execution.

Adversarial tests evaluate direct and indirect prompt injection, role confusion, unsafe escalation, data exfiltration, policy evasion, malicious tool output, and denial-of-service patterns.

Regression tests replay previously observed failures, user complaints, near misses, and corrected trajectories. A resolved defect remains in the suite unless the affected feature is retired.

Longitudinal tests are stable cases rerun after model, harness, tool, policy, or context changes. They estimate whether prompt improvements transfer across configurations.

Portfolio construction follows risk-weighted coverage. Let q_i be the production frequency of scenario i , h_i its harm severity, u_i its uncertainty, and n_i its novelty or change exposure. A prioritization value may be defined as:

$$\text{Priority}_i = a q_i + b h_i + c u_i + d n_i, \quad (3)$$

where weights reflect the deployment context. High-severity cases remain mandatory even when q_i is small.

5.3 Phase 3: Execute Controlled and Repeated Experiments

Offline comparison uses paired tasks: the control and candidate prompts receive the same input, context snapshot, tool definitions, data state, inference parameters, and execution budget. Where mutable tools are involved, each run starts from an isolated or resettable state. All configuration elements are recorded in a run manifest.

Candidate testing should begin with isolated changes. If multiple prompt clauses are modified simultaneously, a combined result cannot identify which change caused an improvement or regression. A practical experiment includes a stable control, one or more isolated candidates, and a combined candidate when interaction effects are important.

Repeated execution is mandatory because a single pass measures capability, not reliability. For task i and k independent runs, SP-CARE reports $\text{pass}@1$, $\text{pass}@k$ when repeated attempts are operationally available, and pass^k when consistent success is required [12]. It also reports within-condition variance and failure-mode diversity.

For online experiments, eligible users or sessions are assigned to stable buckets independently of observed outcomes. Treatment identity is stored with the trajectory but hidden from evaluators. Guardrails define exposure limits, stop conditions, and automatic rollback criteria. Small cohorts are treated as directional screens; confirmatory claims require adequate sample size and pre-specified analysis.

5.4 Phase 4: Assess Through Multiple Evidence Channels

SP-CARE routes each criterion to the strongest available evaluator.

Deterministic validators assess schemas, exact outcomes, state transitions, required confirmations, prohibited tools, test execution, citations, and structured evidence. They are preferred when a rule can be encoded reliably.

Task-specific metrics measure accuracy, completion, groundedness, retrieval performance, state correctness, and domain outcomes.

Trajectory evaluators inspect the sequence of decisions and side effects, including tool choice, arguments, retries, verification, recovery, and unnecessary access.

Calibrated LLM judges evaluate open-ended criteria using atomic rubrics, evidence requirements, blinded variants, swapped pairwise order, and repeated scoring where appropriate.

Human adjudicators label a stratified sample, all critical failures, judge disagreements, and low-confidence cases. Domain experts are required where a criterion depends on legal, medical, financial, safety, or policy interpretation.

User signals include explicit dissatisfaction, corrections, task abandonment, redo behavior, escalation, and direct ratings. These signals are valuable but not assumed to be complete or causal.

Judge calibration uses a human-labeled set that includes positive, negative, ambiguous, and borderline cases. Agreement is reported by criterion rather than only as an overall percentage. Pairwise judges are tested for order



sensitivity, and pointwise judges are tested for scale drift. A judge that performs poorly on a criterion is not used as the sole evaluator for that criterion.

5.5 Phase 5: Release, Monitor, and Revalidate

A release decision combines hard gates, dimension-level thresholds, uncertainty, the risk-adjusted index, and documented trade-offs. Possible outcomes are release, limited rollout, revise and retest, reject, or request additional evidence. Every decision records the prompt version, configuration, dataset version, evaluator versions, result summary, known limitations, approvers, and rollback target.

Production monitoring samples trajectories using the same taxonomy where feasible. New failure clusters enter triage, are traced to a prompt requirement or another system component, and become regression tests after adjudication. Monitoring does not automatically imply that the prompt is the cause; the root cause may be the model, tool, retrieved data, policy layer, or harness.

Revalidation is triggered by:

1. Model or reasoning-configuration changes.
2. Tool additions, removals, schema changes, or permission changes.
3. Context-construction, memory, retrieval, or data-source changes.
4. Policy, regulatory, or product requirement changes.
5. Material prompt edits or interaction among prompt modules.
6. New high-severity incidents or adversarial techniques.
7. Significant drift in user tasks, languages, or session lengths.

Revalidation is tiered. A minor style change may require targeted tests; a model upgrade or permission expansion requires the full high-risk portfolio and representative regression suite.

VI. EVALUATION TAXONOMY AND METRICS

SP-CARE defines ten evaluation dimensions. The taxonomy is deliberately broader than answer quality because system prompts govern both outcomes and behavior.

Table 1. SP-CARE evaluation dimensions, measures, and primary evidence.

Dimension	Construct	Illustrative measures	Primary evidence
Instruction fidelity	Compliance with applicable atomic requirements	requirement pass rate; weighted violation rate; conflict resolution accuracy	deterministic checks; rubric judge; human review
Task effectiveness	Completion of the user's legitimate objective	task success; completeness; terminal-state correctness	task validator; environment state
Claim reliability	Accuracy and support for assertions	factual accuracy; unsupported-claim rate; fabricated-verification rate	evidence matching; human review
Tool behavior	Appropriate and correct use of tools	tool selection; argument validity; unnecessary-call rate; side-effect correctness	trajectory and state logs
Verification discipline	Confirmation before completion claims	test execution; evidence inspection; compile/check success	tool logs; deterministic rules
Safety and security	Resistance to harmful or unauthorized behavior	injection success; privacy breach; unsafe action; authorization violation	adversarial tests; hard gates



Dimension	Construct	Illustrative measures	Primary evidence
Recovery behavior	Effective response to failure and uncertainty	repeated-failure recognition; strategy change; escalation quality	trajectory judge; human review
Robustness and stability	Consistency across perturbations and runs	invariant pass rate; pass ^k ; variance; cross-model transfer	repeated and metamorphic tests
User experience	Observable user outcome and friction	explicit dissatisfaction; corrections; abandonment; satisfaction	conversation signals; surveys
Operational efficiency	Resource use required to achieve quality	latency; tokens; cost; retries; tool-call volume	telemetry

6.1 Requirement-Level Scoring

Let z_{ijr} be the pass indicator for requirement j on task i and run r . A severity-weighted requirement score is:

$$IAR(P) = 1 - [\sum_j v_j (1 - \text{mean}_{i,r}(z_{ijr}))] / \sum_j v_j, \quad (4)$$

where v_j is the requirement severity weight. Critical requirements are additionally represented as hard gates so that their failure cannot be averaged away.

6.2 Task and Trajectory Success

Task success is measured from the terminal environment state whenever possible. Trajectory correctness is separately scored because two agents can reach the same state through different risk profiles. For tool-using tasks, SP-CARE reports:

1. Outcome success rate.
2. Policy-compliant success rate.
3. Side-effect error rate.
4. Tool selection and argument accuracy.
5. Mean and tail tool-call count.
6. Verification-before-claim rate.

6.3 Robustness and Transfer

For a base case x and a meaning-preserving transformation $g(x)$, the invariance violation indicator is one when the relevant behavior changes beyond an allowed tolerance. The Prompt Invariance Rate is:

$$PIR(P) = 1 - [\sum_x 1\{\text{distance}(y_x, y_{g(x)}) > \tau_x\}] / |X|. \quad (5)$$

Cross-model transfer for a prompt change from control P_0 to candidate P_1 is measured through the treatment effect Δ_m on each model m . Transfer consistency can be reported as the fraction of models for which the direction of Δ_m matches the source model and the confidence interval excludes an unacceptable regression.

6.4 User Dissatisfaction

Explicit dissatisfaction is a conservative binary or categorical signal requiring observable evidence such as a correction, complaint, redo statement, or abandonment. Absence of explicit dissatisfaction is not treated as satisfaction. A separate behavioral-issue taxonomy captures failures that may not prompt user feedback.

6.5 Efficiency

Quality and efficiency are reported together. A prompt that adds exhaustive checks to every task may improve verification but impose unacceptable latency or cost. Metrics include median and p95 latency, tokens per successful task, cost per successful task, tool calls per successful task, and retry amplification.

VII. RISK-ADJUSTED SYSTEM PROMPT ASSURANCE INDEX

7.1 Motivation

Average scores can hide severe failures. A prompt that succeeds in 95% of runs but occasionally executes an unauthorized action should not outrank a slightly less capable but safe alternative. Likewise, a prompt whose mean is



high but variance is large may be unsuitable for workflows requiring consistent outcomes. SP-CARE therefore separates non-compensatory gates from a compensatory, risk-adjusted index.

7.2 Definition

For prompt P , let μ_d be the mean normalized score for dimension d , σ_d its execution variability, w_d its decision weight, L the severity-weighted loss distribution, C normalized cost, and T normalized latency. Let $G(P)$ be a gate function equal to zero if any pre-specified critical condition fails and one otherwise. The proposed System Prompt Assurance Index is:

$$\text{SPAI}(P) = G(P) [\sum_d w_d \mu_d - \lambda \sum_d w_d \sigma_d - \rho \text{CVaR}_\alpha(L) - \eta C - \kappa T]. \quad (6)$$

Conditional Value-at-Risk (CVaR) represents expected loss in the worst α -tail of the distribution and is used here as a tail-sensitive decision component [17]. In high-risk contexts, α may be set to 0.95 or 0.99. The coefficients λ , ρ , η , and κ are selected before evaluation based on risk tolerance and operational constraints.

7.3 Hard Gates

Illustrative critical gates include:

1. Unauthorized destructive action or irreversible side effect.
2. Exposure of protected or sensitive data.
3. Successful prompt injection causing prohibited action.
4. Fabricated evidence, test execution, or verification claim.
5. Use of a prohibited tool or privilege path.
6. Failure to preserve required human decision authority.

A gate failure produces an automatic reject or limited-investigation outcome, not a low but compensable score.

7.4 Decision Policy

SPAI should not be used as an unexplained leaderboard. Release requires:

1. All critical gates pass.
2. Each mandatory dimension exceeds its floor.
3. The candidate's paired effect on primary outcomes meets the pre-specified non-inferiority or superiority criterion.
4. No high-severity subgroup shows unacceptable degradation.
5. Cost and latency remain within operational budgets.
6. The decision is robust to reasonable alternative weights.

The dimension vector, confidence intervals, failure taxonomy, and Pareto trade-offs are reported alongside SPAI. Sensitivity analysis identifies whether a decision depends on a narrow weighting choice.

VIII. RESEARCH QUESTIONS AND HYPOTHESES

The empirical study will test the following questions.

RQ1: Does SP-CARE detect more human-confirmed system-prompt regressions than static benchmark evaluation?

H1: SP-CARE will achieve higher recall for high-severity regressions because its portfolio includes boundary, metamorphic, adversarial, regression, and longitudinal tests.

RQ2: Which behavioral dimensions most strongly predict explicit user dissatisfaction and expert release decisions?

H2: Task incompleteness, unsupported claims, failed verification, and repeated unsuccessful actions will be stronger predictors than style deviations.

RQ3: How reliably do system-prompt improvements transfer across model and reasoning configurations?

H3: Prompt treatment effects will exhibit significant prompt-model interaction, and some improvements will not transfer in direction or magnitude.

RQ4: Does calibrated multi-source evaluation align more closely with expert adjudication than a single LLM judge?

H4: Deterministic validators plus calibrated criterion-level judges and targeted human adjudication will achieve higher agreement and lower critical false-negative rates than a single general-purpose judge.

RQ5: Does risk-adjusted ranking expose prompt candidates that appear favorable under average performance but are unsuitable under tail-risk or consistency requirements?

H5: SPAI rankings will differ from mean task-success rankings when candidates have unequal variability, severe tail events, or operational overhead.

RQ6: Does adding production-derived regression cases improve prospective detection of related failures?



H6: Prompt versions evaluated with the continuously updated regression suite will have fewer recurrences of previously observed failure categories than versions evaluated with a fixed benchmark suite.

IX. EMPIRICAL VALIDATION PROTOCOL

9.1 Study Design

The proposed study uses a paired, repeated-measures design. Each task is executed under the same environment across prompt variants and model configurations. The initial pilot validates instruments and estimates variance; the main study tests the hypotheses.

Table 2. Proposed scale of the pilot and main empirical studies.

Study	Tasks	Prompt variants	Model configurations	Repeats	Planned trajectories
Pilot	120	3	2	3	2,160
Main	300	3	3	5	13,500

The three prompt variants are: P0, the stable baseline; P1, a conventional expert revision targeting known issues; and P2, an SP-CARE revision derived from requirement decomposition and risk-based test evidence. The study does not assume that P2 must win; its value is empirically testable.

9.2 Task Sampling

The main portfolio allocates tasks across six families, with severity-stratified sampling:

- 100 representative tasks.
- 40 boundary tasks.
- 40 metamorphic base cases with paired transformations.
- 40 adversarial tasks.
- 40 production-derived or expert-constructed regression tasks.
- 40 longitudinal anchor tasks.

Where production data are unavailable, domain experts construct realistic scenarios using a documented protocol. No private conversation is included without authorization, minimization, and de-identification.

9.3 Configuration Matrix

At minimum, the study compares two model versions or families and two reasoning or inference configurations. Tool definitions, permissions, and environment state are frozen within a paired comparison. A secondary robustness analysis varies context length, irrelevant context, tool failure, and selected user-interaction patterns.

9.4 Baselines

SP-CARE is compared with:

- B0 - Manual spot checking:** a small expert-selected set with informal review.
- B1 - Static benchmark:** representative tasks scored by task success and average quality.
- B2 - Single-judge evaluation:** the static suite plus one uncalibrated general-purpose LLM judge.
- B3 - SP-CARE:** the full portfolio, repeated runs, multi-source evaluation, hard gates, and risk-adjusted decision policy.

Regression-detection ground truth is established through blinded expert adjudication of all candidate failures and a stratified sample of non-failures.

9.5 Judge Development and Calibration

The judge rubric is derived from the requirement register and evaluation taxonomy. Each judge call evaluates one or a small number of related criteria, cites explicit evidence from the trajectory, and returns a structured verdict, confidence, and rationale. Treatment labels, prompt text, and candidate identity are hidden unless required by the criterion.

Calibration uses at least 200 human-labeled trajectories spanning pass, fail, ambiguous, and borderline cases. Twenty percent is reserved as a locked judge-validation set. Metrics include criterion-level precision, recall, F1, balanced



accuracy, calibration error where confidence is used, Cohen's kappa for two raters, and Krippendorff's alpha when more than two raters or missing labels are present.

Critical criteria prioritize recall and are never accepted solely on judge confidence. Pairwise evaluations are repeated with candidate order reversed to estimate position consistency [10]. A judge or criterion is revised when performance falls below its pre-specified floor.

9.6 Data Collection

Each trajectory record contains:

1. Run, task, prompt, model, harness, and dataset identifiers.
2. Timestamp, inference parameters, context hash, and tool-schema hash.
3. Messages, tool calls, observations, state transitions, and terminal state.
4. Deterministic metrics and criterion-level judge outputs.
5. Human labels where sampled.
6. Token usage, latency, cost, retries, and errors.
7. Gate outcomes, severity-weighted failures, and release-decision inputs.

Secrets and sensitive values are excluded or redacted. Raw production content and system prompts may require access controls; reproducibility packages can publish schemas, synthetic examples, hashes, and evaluation code when text cannot be released.

9.7 Primary Outcomes

The primary outcomes are:

1. Recall of human-confirmed high-severity regressions.
2. Critical false-negative rate.
3. Policy-compliant task success.
4. Requirement-level instruction adherence.
5. Judge-human agreement.
6. Repeated-run consistency measured by pass^k and variance.
7. Cross-model transfer consistency.
8. Cost and latency per policy-compliant successful task.

Secondary outcomes include explicit dissatisfaction, recovery behavior, tool efficiency, and failure-mode diversity.

9.8 Statistical Analysis

Binary paired outcomes are compared with McNemar's test and paired effect estimates. Continuous and ordinal metrics use paired bootstrap confidence intervals and suitable paired tests. Mixed-effects logistic regression models success and behavioral failures while accounting for repeated observations:

$$\text{logit Pr}(Y_{ijkl} = 1) = \beta_0 + \beta_1 \text{Prompt}_i + \beta_2 \text{Model}_j + \beta_3(\text{Prompt}_i \times \text{Model}_j) + u_k + v_r, \quad (7)$$

where u_k is a task-level random effect and v_r captures repeated-run or session structure as appropriate.

Subgroup analyses are pre-specified for task family, severity, tool use, context condition, and user pattern. Multiple comparisons are controlled with the Benjamini-Hochberg false discovery rate or a stricter family-wise method for safety-critical hypotheses. Results report confidence intervals and effect sizes, not p-values alone.

SPAI uncertainty is estimated through bootstrap resampling at the task level, preserving repeated runs within each sampled task. Weight-sensitivity analysis recomputes decisions over a plausible weight region. Tail estimates are reported with uncertainty and are not over-interpreted when the number of severe events is small.

9.9 Online Validation

If production experimentation is authorized, an online phase follows offline gating. Stable user or session assignment maps eligible traffic to prompt variants. Exposure begins with a small cohort and expands only when safety and quality monitors remain within bounds. Predefined stop conditions include any critical gate failure, statistically credible deterioration in a primary metric, or material operational instability.

Online evaluation tests ecological validity but does not replace offline tests. User feedback is selectively expressed, treatment exposure may alter task mix, and live tools introduce uncontrolled variation. Analyses therefore record assignment, exposure, environment, and time, and distinguish exploratory monitoring from confirmatory inference.



9.10 Reproducibility Package

The research package should include:

1. Prompt-version hashes and redacted or public prompt text.
2. Requirement-register schema and example records.
3. Test taxonomy, generation protocol, and dataset cards.
4. Tool schemas and resettable environment specifications.
5. Evaluator prompts, model versions, and calibration data.
6. Analysis scripts, decision thresholds, and weighting policy.
7. De-identified trajectory examples and aggregate results.

X. RELEASE GOVERNANCE AND OPERATIONAL INTEGRATION

10.1 Prompt Registry

System prompts are stored in a registry with immutable versions, semantic change descriptions, requirement mappings, compatible configurations, approvers, and rollback references. A change request identifies the diagnosed problem, intended behavior, possible regression, affected requirements, and evidence needed for release.

10.2 Release Gates

The release pipeline applies gates in order:

1. Structural validation of prompt metadata and requirement coverage.
2. Deterministic regression and schema tests.
3. High-risk adversarial and authorization tests.
4. Representative paired evaluation.
5. Judge calibration and human review checks.
6. Risk-index and dimension-floor evaluation.
7. Limited rollout with monitoring, if applicable.

Failure at an early critical gate stops the release. This ordering reduces evaluation cost while preserving safety.

10.3 Continuous Failure Learning

Production monitoring clusters failures by observable behavior rather than vague dissatisfaction alone. Each cluster is triaged to determine whether the primary cause is the prompt, model, tool, context, policy, data, or user-interface layer. Confirmed prompt-related failures produce a requirement update or clarification, a new regression case, and a candidate change. Non-prompt failures remain linked so that the prompt is not used to compensate for a defect that should be fixed elsewhere.

10.4 Roles and Accountability

A minimal governance model includes a prompt owner, evaluation owner, domain reviewer, security or risk reviewer, and release approver. The same person may hold multiple roles in a small team, but critical decisions and judge calibration benefit from independent review. Human authority remains explicit for high-impact application decisions.

XII. THREATS TO VALIDITY

12.1 Construct Validity

The evaluation taxonomy may omit important behaviors or encode them imperfectly. User dissatisfaction is an incomplete proxy because many users do not provide explicit feedback. LLM-judge scores can reflect rubric phrasing, evaluator bias, or superficial cues. Mitigations include requirement traceability, multiple evidence channels, expert validation, criterion-level reporting, and periodic rubric review.

12.2 Internal Validity

Prompt comparisons may be confounded by model updates, non-identical environment state, tool nondeterminism, or context drift. Stable snapshots, paired runs, resettable environments, recorded hashes, and time-bounded experiments reduce these threats. Online A/B assignment does not eliminate interaction effects or changes in traffic composition.

12.3 External Validity

Results from one domain, language, model family, or tool ecosystem may not generalize. The main study therefore includes multiple model configurations, task families, perturbations, and longitudinal anchors. Claims should remain scoped to the evaluated configurations and distributions.



12.4 Conclusion Validity

Small samples, multiple comparisons, noisy human labels, and rare critical events can produce unstable conclusions. Pre-registration, power analysis, paired designs, confidence intervals, multiplicity control, and conservative interpretation of tail estimates are required. A non-significant result does not prove equivalence unless the study is designed for non-inferiority or equivalence testing.

XIII. ETHICAL, PRIVACY, AND SECURITY CONSIDERATIONS

Conversation-based evaluation can expose personal data, proprietary instructions, source code, credentials, or sensitive organizational behavior. Data collection should follow purpose limitation, minimization, access control, retention limits, and de-identification. Human reviewers should see only the information needed for the assigned criterion. Publication artifacts should avoid releasing operational system prompts, attack payloads, or tool details that create disproportionate security risk.

Live experimentation must not expose users to risks that would be unacceptable under the control condition. High-severity safety hypotheses should be tested in sandboxed environments before any production exposure. Automated evaluators should not make unreviewed eligibility, employment, legal, medical, financial, or other high-impact decisions.

XIV. LIMITATIONS AND FUTURE WORK

This manuscript proposes a framework and empirical protocol; it does not report completed SP-CARE experiments. The effectiveness of the requirement decomposition process, portfolio weighting, judge calibration, SPAI formula, and release policy remains to be established through the pilot and main studies. The proposed weights and thresholds are context-dependent and cannot be standardized without evidence from multiple deployments.

Future work should investigate automated requirement extraction with expert verification, active sampling of high-information test cases, causal attribution of prompt failures, evaluator ensembles, domain-specific severity models, multilingual transfer, sequential testing, and privacy-preserving production evaluation. Another important direction is separating defects that should be corrected in deterministic policy or software from behaviors that are appropriately governed through prompts.

XV. CONCLUSION

System prompts increasingly shape the behavior of production AI systems, but their evaluation remains less mature than their operational importance warrants. SP-CARE proposes a continuous, risk-aware approach that treats prompts as versioned behavioral control artifacts. It connects atomic requirements to risk-based tests, controlled repeated execution, multi-source evaluation, hard safety gates, tail-sensitive scoring, release governance, production learning, and model-upgrade revalidation.

The framework's central claim is deliberately testable: compared with static or single-judge approaches, SP-CARE should detect more consequential regressions, better represent reliability and tail risk, and support more defensible release decisions. The proposed pilot and main studies provide the path for accepting, revising, or rejecting that claim. Until those studies are completed, SP-CARE should be understood as a structured research protocol and engineering hypothesis, not as empirically proven superiority.

REFERENCES

1. M. Kim, P. Chapman, S. S. Somarouthu, J. Agrawal, and M. K. Ramanathan, "Continuous Prompt Evaluation: How We Use LLM Judges and Live Signals to Improve Kiro Agent Quality," Kiro, Aug. 21, 2026. Available: <https://kiro.dev/blog/continuous-prompt-evaluation/>
2. P. Liang et al., "Holistic Evaluation of Language Models," arXiv:2211.09110, 2023.
3. M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh, "Beyond Accuracy: Behavioral Testing of NLP Models with CheckList," in Proc. 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 4902-4912, doi: 10.18653/v1/2020.acl-main.442.
4. M. Sclar, Y. Choi, Y. Tsvetkov, and A. Suhr, "Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design or: How I Learned to Start Worrying about Prompt Formatting," in Proc. 12th International Conference on Learning Representations, 2024.



5. K. Zhu et al., "PromptRobust: Towards Evaluating the Robustness of Large Language Models on Adversarial Prompts," arXiv:2306.04528, 2024.
6. T. P. Zollo, T. Morrill, Z. Deng, J. C. Snell, T. Pitassi, and R. Zemel, "Prompt Risk Control: A Rigorous Framework for Responsible Deployment of Large Language Models," in Proc. 12th International Conference on Learning Representations, 2024.
7. L. Zheng et al., "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena," in Advances in Neural Information Processing Systems, Datasets and Benchmarks Track, 2023.
8. Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, and C. Zhu, "G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment," in Proc. 2023 Conference on Empirical Methods in Natural Language Processing, 2023, pp. 2511-2522, doi: 10.18653/v1/2023.emnlp-main.153.
9. H. Hashemi, J. Eisner, C. Rosset, B. Van Durme, and C. Kedzie, "LLM-Rubric: A Multidimensional, Calibrated Approach to Automated Evaluation of Natural Language Texts," arXiv:2501.00274, 2024.
10. L. Shi, C. Ma, W. Liang, W. Ma, and S. Vosoughi, "Judging the Judges: A Systematic Investigation of Position Bias in Pairwise Comparative Assessments by LLMs," arXiv:2406.07791, 2024.
11. X. Liu et al., "AgentBench: Evaluating LLMs as Agents," in Proc. 12th International Conference on Learning Representations, 2024.
12. S. Yao, N. Shinn, P. Razavi, and K. Narasimhan, "Tau-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains," arXiv:2406.12045, 2024.
13. E. Debenedetti, J. Zhang, M. Balunovic, L. Beurer-Kellner, M. Fischer, and F. Tramer, "AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents," in Advances in Neural Information Processing Systems, Datasets and Benchmarks Track, 2024.
14. K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection," arXiv:2302.12173, 2023.
15. B. Wang et al., "DecodingTrust: A Comprehensive Assessment of Trustworthiness in GPT Models," in Advances in Neural Information Processing Systems, Datasets and Benchmarks Track, 2023.
16. C. Autio, R. Schwartz, J. Dunietz, S. Jain, M. Stanley, E. Tabassi, P. Hall, and K. Roberts, Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile, NIST AI 600-1, National Institute of Standards and Technology, 2024, doi: 10.6028/NIST.AI.600-1.
17. R. T. Rockafellar and S. Uryasev, "Optimization of Conditional Value-at-Risk," Journal of Risk, vol. 2, no. 3, pp. 21-42, 2000, doi: 10.21314/JOR.2000.038.
18. Kumar, S. N. P. (2026). Advanced architectural frameworks for scalable, production-grade agentic RAG pipelines. International Journal of Research and Applied Innovations, 9(1), 13491–13498. <https://doi.org/10.15662/IJRAI.2026.0901001>
19. Kumar, S. N. P. (2025). A secure accountability framework for multi-modal agent systems: Detecting, mitigating, and auditing data-poisoning attacks via Model Context Protocol servers. Journal of Computer Science and Technology Studies, 7(12), 1–5. <https://doi.org/10.32996/jcsts.2025.7.12.1>
20. Kumar, S. N. P. (2025). Building scalable and reliable agentic AI systems: A technical blueprint for autonomous intelligence. Global Journal of Engineering and Technology Research, 1(3). <https://doi.org/10.65150/ep-gjetrv1e3/2025-07>
21. Kumar, S. N. P. (2025). Regulating autonomous AI agents: Prospects, hazards, and policy structures. Journal of Computer Science and Technology Studies, 7(10), 393–399. <https://doi.org/10.32996/jcsts.2025.7.10.41>
22. Kumar, S. N. P. (2025). Hallucination detection and mitigation in large language models: A comprehensive review. Journal of Information Systems Engineering and Management, 10(60S), 435–443. <https://doi.org/10.52783/jisem.v10i60s.13133>
23. Kumar, S. N. P. (2025). Multi-agent AI systems in finance: Models, applications, and challenges. International Journal of Advanced Research in Computer Science & Technology, 8(1), 11555–11573. <https://doi.org/10.15662/IJARCST.2025.0801006>
24. Kumar, S. N. P. (2025). Scalable cloud architectures for AI-driven decision systems. Journal of Computer Science and Technology Studies, 7(8), 416–421. <https://doi.org/10.32996/jcsts.2025.7.8.46>
25. Kumar, S. N. P. (2025). Recent innovations in cloud-optimized retrieval-augmented generation architectures for AI-driven decision systems. European Modern Studies Journal, 9(4), 870–883. [https://doi.org/10.59573/emsj.9\(4\).2025.81](https://doi.org/10.59573/emsj.9(4).2025.81)
26. Kumar, S. N. P., Gangurde, R., & Mohite, U. L. (2025). RMHAN: Random multi-hierarchical attention network with RAG-LLM-based sentiment analysis using text reviews. International Journal of Computational Intelligence and Applications, 24(4). <https://doi.org/10.1142/S1469026825500075>
27. Kumar, S. N. P. (2025). Ethical frameworks for AI-driven decision systems: A comprehensive analysis. Global Journal of Computer Science and Technology, 25(1), 53–60. <https://doi.org/10.34257/gjcsdvol25is1pg53>



28. Kumar, S. N. P. (2025). Real-time justice intelligence: AI-enabled data governance transforming city criminal justice. *Global Journal of Engineering and Technology Research*, 1(3). <https://doi.org/10.65150/ep-gjetr/v1e3/2025-08>
29. Kumar, S. N. P. (2025). Fraud detection in banking using generative AI. *Sarcouncil Journal of Engineering and Computer Sciences*, 4(11), 133–145. Publisher record
30. Kumar, S. N. P. (2025). AI and cloud data engineering transforming healthcare decisions. *Sarcouncil Journal of Engineering and Computer Sciences*, 4(8), 76–82. Publisher record
31. Kumar, S. N. P. (2025). Quantum-enhanced AI decision systems: Architectural approaches for cloud-based machine learning applications. *Sarcouncil Journal of Multidisciplinary*, 5(8). Publisher record
32. Kumar, S. N. P. (2025). Navigating the AI horizon: Transformations, ethical imperatives, and pathways to responsible innovation. *Sarcouncil Journal of Applied Sciences*, 5(10), 34–43. Publisher record
33. Choudhury, A., Balasubramaniam, S., Kumar, A. P., & Kumar, S. N. P. (2025). PSSO: Political squirrel search optimizer-driven deep learning for severity-level detection and classification of lung cancer. *International Journal of Information Technology & Decision Making*, 24(8), 2373–2406. <https://doi.org/10.1142/S0219622023500189>
34. Preetham, A., Vyas, S., Kumar, M., & Kumar, S. N. P. (2024). Optimized convolutional neural network for land-cover classification via improved lion algorithm. *Transactions in GIS*, 28(4), 769–789. <https://doi.org/10.1111/tgis.13150>
35. Mahalakshmi, T., Beevi, S. Z., Navaneethakrishnan, M., Ramya, P., & Kumar, S. N. P. (2024). Optimized attention-driven bidirectional convolutional neural network for Facebook sentiment classification. *International Journal of Business Data Communications and Networking*, 19(1), 1–20. <https://doi.org/10.4018/IJBDCN.349572>
36. Choudhury, A., Vuppu, S., Singh, S. P., Kumar, M., & Kumar, S. N. P. (2023). ECG-based heartbeat classification using exponential-political optimizer-trained deep learning for arrhythmia detection. *Biomedical Signal Processing and Control*, 84, 104816. <https://doi.org/10.1016/j.bspc.2023.104816>
37. Masthan, M., Pazhanikumar, K., Chavan, M., Mandala, J., & Kumar, S. N. P. (2023). SCSLnO-SqueezeNet: Sine cosine-sea lion optimization-enabled SqueezeNet for intrusion detection in IoT. *Network: Computation in Neural Systems*, 34(4), 343–373. <https://doi.org/10.1080/0954898X.2023.2261531>
38. Nalavade, J. E., Kolli, C. S., & Kumar, S. N. P. (2023). Deep embedded clustering with matrix factorization-based user-rating prediction for collaborative recommendation. *Multiagent and Grid Systems*, 19(2), 169–185. <https://doi.org/10.3233/MGS-230039>
39. Boopathi, M., Chavan, M., Jebanazer, J. J., & Kumar, S. N. P. (2023). An approach for DoS attack detection in cloud computing using sine cosine anti-coronavirus optimized deep Maxout network. *International Journal of Pervasive Computing and Communications*, 19(5), 666–688. <https://doi.org/10.1108/IJPCC-05-2022-0197>
40. Kumar, S. N. P. (2022). Improving fraud detection in credit-card transactions using autoencoders and deep neural networks [Doctoral dissertation, The George Washington University]. GW ScholarSpace. https://scholarspace.library.gwu.edu/concern/gw_etds/cv43nx607
41. Kumar, S. N. P. (2022). Text classification: A comprehensive survey of methods, applications, and future directions. *International Journal of Technology, Management and Humanities*, 8(3), 39–49. <https://doi.org/10.21590/ijtmh.8.03.04>
42. Kumar, S. N. P. (2022). Machine learning regression techniques for modeling complex industrial systems: A comprehensive summary. *International Journal of Humanities and Information Technology*, 4(1–3), 67–79. <https://doi.org/10.21590/ijhit.06.01-3.07>