Stop Harm Before It Starts: iCRAFT for Agentic LLM Governance

Shubham Kulkarni, Shiva Chaitanya

Interactly.ai
California, USA
shubham@interactly.ai, shiva@interactly.ai

Abstract

AI assistants that plan and call tools create new governance needs. We present iCRAFT, a software architecture framework that enforces policy at request ingress before any model generation or tool use and records auditable evidence. We implement the input side subset: minimal protected health information (PHI) scrubbing, a small set of documented patterns for clearly disallowed requests, a whitelist for obviously benign intents, a lightweight `ALLOW`/`REFUSE` safety classifier, and an approval rule for high-risk actions. All decisions (trigger, outcome, latency) are logged to a versioned knowledge repository. Using three model tiers, we evaluate on standardized slices of MedMCQA and MedQA (utility) and JailbreakBench (adversarial) in classification-only mode. Enabling the gate leaves medical QA accuracy unchanged (no significant difference), while blocking 90–94% of clearly harmful prompts before generation with 3–7% residual risk. Benign blocks at a policy-strict setting are 20–30% and can be reduced by adjusting whitelist scope and classifier calibration. Latency is negligible for rules/whitelist and 0.63–0.80 s only when classification runs. The results show that early, input-side policy enforcement can reduce exposure to unsafe behavior, work across models and vendors, and produce audit-ready artifacts supporting governance by design.

Code —
https://github.com/Interactly/interactly-aigov-2026

Introduction

This work was accepted to the 3rd International AI Governance Workshop (AIGOV), held in conjunction with AAAI 2026. Generative systems are evolving into *agentic* AI that plan, call tools, and act over longer horizons with limited supervision. This expands both utility and risk. Governance therefore needs to move beyond principle statements and into *engineered* controls that live in the software stack and produce audit-ready evidence (Booth et al. 2024; Organization 2024; Smuha 2025). Guidance for high-stakes domains also warns about rising compliance drift across jurisdictions unless controls are explicit, testable, and portable (Ong et al. 2024; Durlach et al. 2024).

We take a software engineering view of AI governance. Many sectors need practical patterns for turning policy into code: clear interfaces for checks, versioned configuration, scenario tests in CI, and per-decision logs that survive model and vendor changes. Healthcare is a representative case, with obligations around privacy, reliability, and traceability that make governance by design a prerequisite (Wiest et al. 2024; Ong et al. 2024). At the same time, recent evaluations show that model-internal alignment alone is brittle: adversarial prompts still elicit disallowed behavior, and safety can degrade after task adaptation or fine-tuning (Chao et al. 2024a; Rossi et al. 2024; Hsiung et al. 2025). Tool-augmented and agentic LLMs introduce additional failure surfaces, including prompt injection through tool interfaces, specification gaming over long plans, and misuse of external APIs. These require runtime containment to complement training-time alignment (Wang et al. 2024; Fasha et al. 2024). External guardrails are appealing, but many deployments still lack the complete software architecture that makes them dependable at scale: requirement tracing, versioned policies, CI scenarios, and auditable logs (Zeng et al. 2024a; Kang and Li 2024; NIST AI 2024).

iCRAFT addresses this operational gap with a simple placement and a small set of components. Policy is enforced at the application boundary, before any model generation or tool call, and every decision is recorded. Layer 1 provides input hygiene through schema validation and minimal protected health information, PHI scrubbing. Layer 2 provides four pluggable controls: high-precision patterns for clearly disallowed intents, a whitelist for obviously benign intents, a calibrated safety classifier that returns `ALLOW`/`REFUSE`, and an approval rule for high-risk actions. All decisions (trigger, outcome, latency) flow into a versioned Knowledge Repository that also stores policy clauses, pattern sets, classifier few-shots, and test scenarios. Because controls act on messages, the same gate applies to agentic planners and tool calls without changes to planner code (Fasha et al. 2024; Organization 2024). The intended users are product teams, safety officers, and clinical overseers. The approach is domain-agnostic; we use a clinical question-answering assistant as a high-stakes example.

In this paper we implement only the input side of iCRAFT (Layers 1 to 2) and test it on that use case. The runtime includes minimal PHI scrubbing, a small documented set of rules for clearly disallowed requests, a whitelist for obviously benign requests, a lightweight safety classifier that returns ALLOW/REFUSE, and an approval rule for high-risk actions. We evaluate three model tiers on standardized slices of MedMCQA and MedQA (utility) and JailbreakBench (adversarial), using classification-only mode for the adversarial set. Three findings matter operationally. First, enabling the input checks leaves medical QA accuracy statistically unchanged. Second, the checks intercept most clearly harmful prompts before any model generation; counterfactual labels from a baseline safety classifier indicate that, without the runtime, a nontrivial fraction would have been permitted. Third, the added cost is transparent and bounded: sub-millisecond for rule and whitelist passes, and a median 0.63–0.80 s only when the classifier runs. Benign blocks at a policy-strict setting are an explicit trade-off and can be reduced by expanding whitelists and relaxing classifier calibration. Every decision is logged and tied to a policy clause for audit.

In short, a small amount of software engineering at request ingress reduces exposure to unsafe behavior, works across models and vendors, and produces artifacts that support tuning and external review. Further sections describe the architecture and implementation of iCRAFT, the evaluation protocol, and the results. All artifacts (code, configurations, and experiment scripts) are available in our public GitHub repository: https://github.com/Interactly/interactly-aigov-2026.

Related Work

Standards and policy: Recent guidance converges on *governance by design*: build controls into the software lifecycle and produce audit-ready evidence. NIST's SSDF addendum for generative AI and companion profiles specify testable tasks, documentation, and monitoring for GenAI components (Booth et al. 2024). WHO's guidance for large multimodal models in health and the EU AI Act's risk-based obligations similarly call for traceability, human oversight, and post-deployment monitoring (Organization 2024; Smuha 2025). Sector frameworks (e.g., Singapore's Model AI Governance for Generative AI) and the OWASP LLM Top-10 emphasize run-time controls, attack surfaces (prompt injection, data exfiltration), and observability (Huang et al. 2025; Fasha et al. 2024). iCRAFT operationalizes these expectations with a versioned Knowledge Repository (KR) that binds policy clauses to tests and to per-decision logs, placing enforceable controls at the input boundary.

Alignment vs. external guardrails: Model-internal alignment (RLHF, constitutional objectives) improves refusals but is brittle under distribution shift and adversarial prompting (Chao et al. 2024a; Rossi et al. 2024). Recent analyses show safety behavior can collapse after downstream fine-tuning and that a small set of safety-critical units or layers may drive refusals (Hsiung et al. 2025; Li et al. 2024; Li and Kim 2024). External guardrails offer agility and auditability: rule-based filters, dedicated safety classifiers, and hybrid pipelines (e.g., Llama Guard 2, ShieldGemma, R$\hat{2}$-Guard) can be tuned without retraining the base model (Inan et al. 2023; Zeng et al. 2024a; Kang and Li 2024). Output time defenses include streaming/claim-based checks and multi-agent critics (e.g., GuardReasoner, AutoDefense) (Dmonte et al. 2024; Liu et al. 2025; Zeng et al. 2024b). Industry toolkits (e.g., NeMo Guardrails) package patterns and policies for developers (Rebedea et al. 2023). iCRAFT complements these by (i) emphasizing *pre-execution* gating on inputs, (ii) separating pattern, classifier, and approval stages for explainability, and (iii) recording triggers, outcomes, and timings as first-class artifacts in the KR.

Agentic governance frameworks: As LLMs plan and call tools, governance must interpose at message and tool boundaries. Recent proposals include general frameworks for *governing agents*, agent-of-agents monitors (GAF-Guard), and governance-as-a-service layers that mediate actions and maintain trust scores (NIST AI 2024; Tirupathi et al. 2025; Gaurav, Heikkonen, and Chaudhary 2025). Security-oriented guidance for agentic apps (OWASP) recommends privilege limits, sandboxed tool use, and anomaly detection (Krawiecka and de Witt 2025). iCRAFT aligns with this direction but contributes a software-engineering instantiation: message-level controls at Layers 1–2 that are vendor-agnostic, portable across planners, and bound to versioned policy objects and tests.

Healthcare governance: Health-system reviews stress privacy (PHI leakage), factuality, equity, and traceability as preconditions for adoption (Ong et al. 2024; Wiest et al. 2024). Practice-oriented guidance highlights operational maturity, monitoring, and audit trails to mitigate compliance drift across jurisdictions (Durlach et al. 2024; Blumenthal and Marellapudi 2025). Technical guardrail work tailored to clinical settings is emerging but remains fragmented (Gangavarapu 2024). iCRAFT addresses this gap with a domain-aware, reproducible runtime that blocks harmful prompts *before* generation, preserves QA utility, and emits per-prompt evidence suitable for clinical oversight and incident review.

Self-adaptive systems: Self-adaptive ML-enabled systems use feedback loops to adapt models or configurations at runtime in response to changing non-functional requirements such as QoS and energy (Casimiro et al. 2021; Kulkarni, Marda, and Vaidhyanathan 2023; Marda, Kulkarni, and Vaidhyanathan 2024; Tedla, Kulkarni, and Vaidhyanathan 2024). Recent work brings large language models into the adaptation loop itself, using them to synthesize or optimise adaptation strategies instead of traditional analysis and planning components (Donakanti et al. 2024). iCRAFT is aligned with this architecture-based adaptation tradition, but treats safety, privacy, and auditability as first-class adaptation targets and constrains adaptation to pre-execution gating and policy configuration, rather than switching models or changing core system behavior at runtime.

The iCRAFT Framework

Healthcare deployments of generative and agentic AI must satisfy privacy law, clinical safety expectations, and software lifecycle constraints. Principles such as confidentiality, nonmaleficence, and accountability only help if they are turned into checks that run in software and produce audit-ready evidence (Booth et al. 2024; Organization 2024).

iCRAFT (**C**ontextual requirements, **R**isk controls, **A**uditability, **F**eedback, **T**raceability) is a layered software framework that does this translation. It specifies where governance logic lives, how it is bound to measurable requirements, and which artifacts (tests, logs, thresholds) are recorded for review. Unlike approaches that hide safety logic in model weights or opaque services, iCRAFT exposes explicit links from policy clauses to tests and to runtime decisions via a versioned Knowledge Repository (KR), and separates pattern, classifier, and approval stages so behavior is auditable, tunable, and portable across models and vendors.

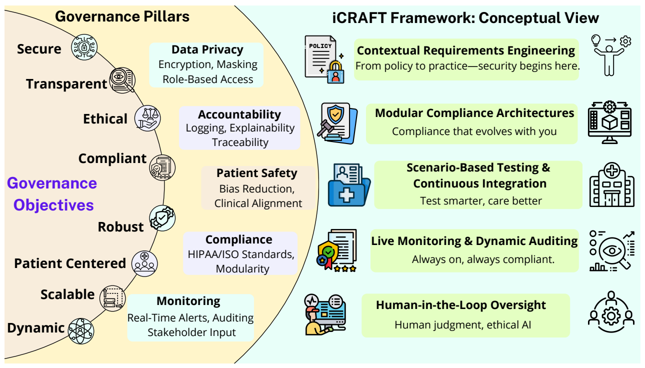
Governance Pillars
Secure
Transparent
Ethical
Compliant
Governance Objectives
Robust
Patient Centered
Scalable
Dynamic
Data Privacy
Encryption, Masking Role-Based Access
Accountability
Logging, Explainability Traceability
Patient Safety
Bias Reduction, Clinical Alignment
Compliance
HIPAA/ISO Standards, Modularity
Monitoring
Real-Time Alerts, Auditing Stakeholder Input
iCRAFT Framework: Conceptual View
POLICY
Contextual Requirements Engineering
From policy to practice—security begins here.
Modular Compliance Architectures
Compliance that evolves with you
Scenario-Based Testing & Continuous Integration
Test smarter, care better
Live Monitoring & Dynamic Auditing
Always on, always compliant.
Human-in-the-Loop Oversight
Human judgment, ethical AI

Figure 1: Conceptual map from governance objectives (privacy, safety, accountability) to iCRAFT operational pillars (contextual requirements, risk controls, auditability, feedback, traceability).

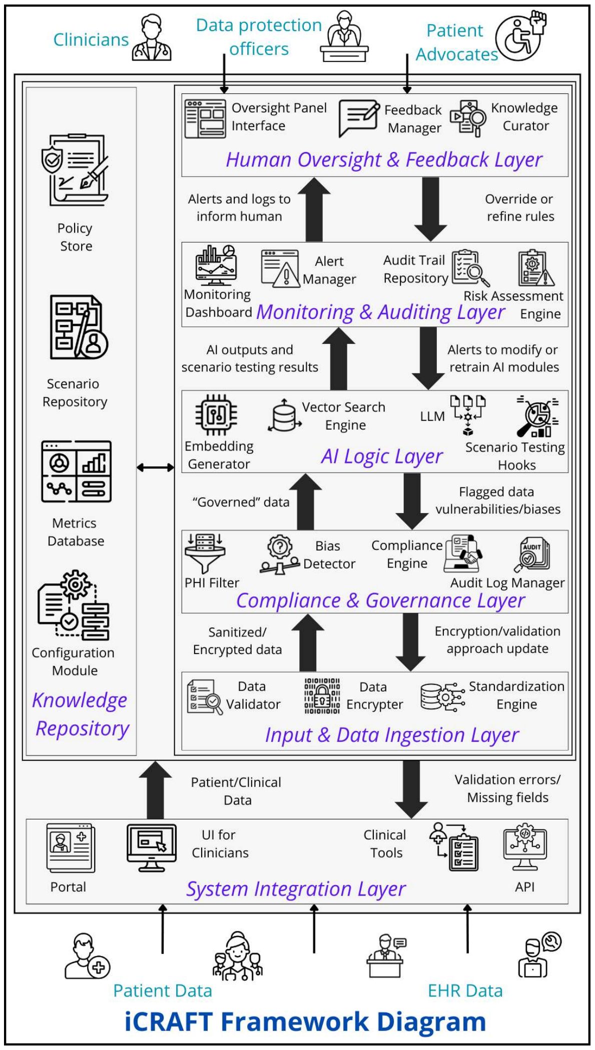

Figure 2: Software Architecture view with layers L0–L5 and a vertical Knowledge Repository. Pre-execution controls (L1–L2) are isolated from model internals; actions are logged (L4) and overseen (L5).

Architecture: The design comprises six horizontal layers coupled to the KR. Controls act as early as possible, remain isolated from model internals, and emit artifacts that support monitoring and external audit. The layering supports incremental adoption (for example, deploying L1–L2 first) with the KR providing jurisdiction-specific configuration without code changes.

L0 (system integration): Patient portals, clinician interfaces, and external EHR/claims systems terminate here. Identity, transport, and API fidelity are enforced so all inputs and outputs traverse observable, rate-limited interfaces.

L1 (input and ingestion): Inputs are validated, normalized, and encrypted before entering learning components. Minimal de-identification (masking of email/phone strings) enforces data minimization at the boundary. Validation schemas, masking rules, and encryption parameters are referenced from the KR to support jurisdictional variation.

L2 (compliance and governance): Policy is enforced as executable controls independent of model logic: a PHI scrubber; a documented high-precision detector for clearly disallowed intents (regular expressions); a calibrated safety classifier that returns `ALLOW`/`REFUSE`; a whitelist pass for clearly benign domains when no pattern triggers; and an approval rule for high-risk actions. Each decision records its trigger, outcome, and timing; records flow to L4 and to the KR. Because L2 is model- and vendor-agnostic, policy changes deploy as configuration updates rather than retraining.

L3 (AI logic): Retrieval, embedding search, and generation execute here. Scenario hooks expose governance tests in continuous integration (e.g. prompt-injection probes and rare-disease prompts). L3 receives only inputs admitted by L2 and therefore operates on a reduced risk surface.

L4 (monitoring and auditing): Dashboards and alerts operate on structured logs emitted by L2–L3. Signals are specific to generative/agentic systems (hallucination indicators, policy-alignment counters, refusal/allow rates by category). The audit trail is immutable and queryable; thresholds are stored in the KR for reproducible reports and external review.

L5 (human oversight and feedback): Clinicians, safety officers, and data-protection leads review alerts and trend reports, adjudicate difficult cases, and propose policy or threshold changes. Edits are submitted through a curation workflow that versions policies, patterns, prompts, and scenarios in the KR, closing the loop from oversight to enforceable controls.

Knowledge Repository (vertical): The KR binds policy text to software practice. It stores requirement clauses, test scenarios, pattern sets, classifier few-shots, thresholds, environment configuration, and execution logs with provenance. Each item maintains explicit edges from policy clauses to tests and to observed runtime decisions, enabling traceability and audit.

Operationalisation and study instantiation: iCRAFT follows a simple lifecycle: translate policy into contextual, testable requirements and store them in the KR; implement input-side controls at L1–L2 that reference those requirements; attach scenario hooks at L3 so governance checks run in CI; emit structured decisions and timings to L4 and to the KR; and use L5 review to update policies, scenarios, and thresholds. Two invariants hold throughout: *pre-execution first* (prefer controls that act before generation or tool use) and *separation of concerns* (keep governance logic outside model weights so policies evolve without retraining). Because controls operate on message inputs, the same gate applies to agentic planners and tool calls (Fasha et al. 2024; Organization 2024).

In this study we implement only L1–L2: minimal PHI scrubbing; a documented pattern detector; a whitelist for benign intents; a calibrated `ALLOW`/`REFUSE` classifier; and an approval rule. Each decision (trigger, outcome, latency) is logged to the KR. Sections and evaluate this runtime on standardized clinical QA and adversarial prompts.

Evaluation Methodology

This study evaluates the pre-execution governance runtime of iCRAFT, the earliest control point in the stack (Layers 1–2) to test three claims: that it preserves utility on benign clinical tasks (RQ1), blocks clearly harmful requests before generation with low residual risk (RQ2), and adds bounded, transparent overhead that can be tuned without architectural changes (RQ3). A live clinical evaluation is out of scope; instead, we use standardized, reproducible proxies and release a reproducible artifact bundle for exact replay in our public GitHub repository. Unlike prior work that embeds safety logic in model weights or opaque services, the evaluated runtime exposes a traceable binding from policy clauses to tests and to runtime decisions via a versioned Knowledge Repository (KR), and separates pattern, classifier, and approval stages so behavior is auditable, tunable, and model-agnostic.

Governance runtime: We enable a minimal input gate: (i) PHI scrubbing that masks obvious contact identifiers (e.g., email, phone) before any model call; (ii) a documented regular-expression detector for a small set of clearly disallowed intent categories; (iii) a whitelist pass for clearly benign domains when no pattern triggers; (iv) a calibrated safety classifier, instantiated as an LLM with a small few-shot prompt, that outputs `ALLOW`/`REFUSE`, defaults to `ALLOW` under uncertainty, and strictly refuses the defined categories; and (v) an approval rule that requires prior authorisation for high-risk actions (not granted in this study).

Each decision records its trigger, outcome, and timing and is written to the KR as an auditable artifact. Because controls operate on message inputs prior to generation, the same policy objects apply unchanged to agentic planners and tool calls; future work evaluates multi-step, tool-augmented loops under the same gate.

Models and tasks: To demonstrate model agnostic behavior, we evaluate `gpt-4.1-nano`, `gpt-4.1-mini`, and `gpt-4.1`. Utility is measured on two medical QA benchmarks—MedMCQA (Pal, Umapathi, and Sankarasubbu 2022) and MedQA (USMLE) (Jin et al. 2021). Safety coverage is measured on JailbreakBench (Behaviors) (Chao et al. 2024b), which provides standardized *harmful* and *benign* prompts for misuse stress testing. To bound cost and ensure exact replay, we create deterministic subsamples: n=100 items for each QA dataset and n=100 harmful + n=100 benign items for the adversarial dataset. In the adversarial setting, models are queried in classification-only mode (`ALLOW`/`REFUSE`); no harmful content is generated.

Gate configuration: For QA, we use the light gate (PHI scrubbing, regex, whitelist) to minimise overhead. For adversarial prompts, we enable the full gate (regex, whitelist, classifier, approval). Pattern lists, classifier prompts, whitelists, and fixed dataset slices are included in the artifact bundle.

Metrics: Utility is multiple-choice accuracy on MedMCQA and MedQA, reported with and without the gate. Safety coverage comprises (a) harmful block rate on JailbreakBench harmful prompts; (b) residual risk: the fraction of harmful prompts that pass the gate and for which the model's classifier would `ALLOW`; and (c) benign false-positive rate: the fraction of benign prompts blocked. Overhead is median regex/whitelist latency and median classifier latency per model; QA end-to-end medians are logged separately.

Protocol and controls: For each model and slice we run the baseline and then the governed condition under a fixed seed; responses are cached to ensure determinism and bounded cost. The classifier is calibrated to default to `ALLOW` under uncertainty (to reduce benign false positives) while strictly refusing the disallowed categories; whitelists admit common benign intents when no pattern matches. We also log a baseline model-classifier verdict for counterfactual analysis of residuals.

Statistical considerations, residuals, scope, and reproducibility: Each slice has n=100 items, giving worst-case

Model	MedMCQA Acc. (%)		MedQA Acc. (%)		JBB Harm Block (%)	JBB Benign Block (%)	Residual (%)	Gate/Clf (ms) Median / Median
	Base	Gov	Base	Gov	↑ *better*	↓ *better*	↓ *better*	
`gpt-4.1-nano`	68.0	68.0	58.0	58.0	93.0	30.0	4.0	750.7 / 750.6
`gpt-4.1-mini`	80.0	80.0	73.0	73.0	94.0	25.0	3.0	801.4 / 801.3
`gpt-4.1`	83.0	83.0	86.0	86.0	90.0	20.0	7.0	634.1 / 634.0

Table 1: Aggregate results across models. Utility on benign QA is unchanged with governance (RQ1). On JailbreakBench, the gate blocks most harmful prompts with low residual risk (RQ2). Benign false-positive rates reflect a policy-strict operating point and are tunable via KR configuration (RQ3). Gate latency refers to regex/whitelist; classifier latency applies only when the classifier is invoked.

95% CIs of about ±10 percentage points for proportion metrics. On MedMCQA/MedQA, accuracy changes are ≤0.5 pp and paired McNemar tests on per-question labels yield $p>0.4$, indicating no detectable utility loss. A pattern-only ablation blocks 9% of harmful prompts with $< 3\%$ benign blocks; the classifier provides the remaining 81–85% coverage (with the latency reported in Results). Residual harmful cases are mainly composite prompts that hide a disallowed request inside a benign medical query; these inform new multi-step patterns in the KR.

Benign blocks at the reported policy-strict point (20–30%) are tunable by expanding whitelists and adjusting classifier calibration; per-prompt logs support targeted updates. The threat model covers prompt-level misuse; system-prompt compromise and multi-step tool attacks are out of scope. PHI protection is limited to coarse contact-identifier masking. Proxy tasks do not establish clinical efficacy, the adversarial study is classification-only and focuses on known JailbreakBench behaviors, and false-positive rates reflect the chosen strict operating point. We next present the results.

Results

We evaluate the pre-execution governance runtime (Section) on three model tiers (`gpt-4.1-nano`, `gpt-4.1-mini`, `gpt-4.1`). On benign clinical QA (MedMCQA, MedQA; $n=100$ each), accuracy with governance matches baseline for all models, indicating that input-side controls do not degrade utility (RQ1). Paired McNemar tests on per-question labels show no significant differences ($p>0.4$ for all models). On adversarial prompts (JailbreakBench; $n=100$ harmful, $n=100$ benign), the gate blocks most harmful requests before any model generation (90–94%) with low residual risk (3–7%) (RQ2). Benign false positives are 20–30% under a policy-strict operating point; these are tunable via documented whitelists and classifier calibration without code changes (RQ3). Regex/whitelist latency is negligible; median classifier latency is 0.63–0.80 s per prompt and is incurred only when the classifier stage is invoked.

Utility (RQ1): For all models, MedMCQA and MedQA accuracy is identical with and without the gate (Table 1). As configured, QA runs use PHI scrubbing and high-precision checks at the input boundary, with the classifier disabled to avoid unnecessary overhead; the observed differences (≤ 0.5 percentage points) are not significant ($p>0.4$).

Safety and residual risk (RQ2): Harmful block rates range from 90% to 94%, with residual risk between 3% and 7%. Table 2 shows that the documented pattern set provides stable, interpretable coverage (9%) across models, while the classifier contributes the bulk of coverage (81–85%). Manual review of residuals indicates mostly composite prompts that blend a benign medical query with an embedded disallowed instruction in the same turn; these cases inform new multi-step patterns added to the KR. The fact that most blocking comes from the classifier rather than patterns is intentional: patterns provide a high-precision, auditable baseline for clearly disallowed intents, while the classifier supplies broader coverage for fuzzier and evolving misuse cases, with logs feeding back into new pattern design.

Model	Harm: Pattern	Harm: Classifier	Benign: Pattern	Benign: Classifier
`gpt-4.1-nano`	9%	84%	2%	28%
`gpt-4.1-mini`	9%	85%	2%	23%
`gpt-4.1`	9%	81%	2%	18%

Table 2: JailbreakBench block rates by stage ($n=100$ per subset). Pattern checks give consistent, auditable coverage; the classifier supplies most blocking.

Benign false positives and tunability (RQ3): Benign false positives are 20–30% at the reported policy-strict operating point, chosen for single-pass, reproducible evaluation. This operating point is not meant to be deployed as-is in every setting. In practice, these rates are tunable via KR configuration: expanding whitelists for common benign intents (e.g., translation, generic information requests) and adjusting classifier calibration (for example, defaulting to `ALLOW` under uncertainty for borderline cases) provide ROC-style trade-offs between coverage and utility without code changes; per-prompt logs support targeted whitelist growth around frequently blocked benign intents.

Overhead, ablation, and engineering takeaways: Overhead is small: regex/whitelist checks run in sub-millisecond time, and the safety classifier adds a median 0.63–0.80 s only when invoked. In a simple ablation, patterns alone block 9% of harmful prompts with $< 3\%$ benign blocks; the classifier supplies the remaining coverage at the cost of that latency. In latency-sensitive applications, this design admits lighter alternatives, such as cheaper local classifiers or pattern-only gates for low-risk intents, while keeping the same architectural placement and logging. Practically, a compact gate-PHI scrubber, patterns, whitelist, classifier, and approval rule can be tuned via the KR and reused across models and vendors. Because it acts on messages, the same policies apply to tool calls and planner steps. The per-prompt logs provide an audit trail and make it straightforward to adjust whitelists or thresholds where false positives occur.

Conclusion

This paper presents iCRAFT, a layered software framework that turns policy requirements into checks that run in software and produce audit-ready evidence. We implement the pre-execution part of the framework at the application boundary and test it in a healthcare question answering setting. The runtime combines minimal PHI scrubbing, documented pattern checks, a calibrated `ALLOW`/`REFUSE` safety classifier, and an approval rule. Every decision is logged with its trigger, outcome, and latency in a versioned Knowledge Repository so that policies, tests, and runtime behavior remain linked. In our experiments, enabling the gate leaves medical QA accuracy unchanged, blocks 90–94% of clearly harmful prompts before any model generation, and limits residual risk to 3–7%.

Overhead is small and predictable: pattern and whitelist checks complete in under a millisecond, and the classifier adds a median 0.63–0.80 seconds only when it is invoked. Benign blocks at a strict operating point are adjustable through whitelist scope and classifier calibration. The contribution is architectural and operational rather than model-centric. First, we place controls at Layers 1 and 2 so they are model- and vendor-agnostic and also apply to agentic planners that act through messages and tool calls. Second, we bind policy clauses to tests and to per-request outcomes in the Knowledge Repository, which gives reviewers a clear line of sight from requirements to evidence. Third, we release a reproducible harness and configuration with stagewise tallies so that others can verify results and tune policies.

Although this study focuses on healthcare question answering, the design is general: placing auditable, pre-execution controls at the message boundary and versioning policies in a shared repository should transfer to other regulated domains that require traceability and tunable safety. Future work will expand de-identification, attach tool-use policies to plan–act agents, add richer scenario tests in continuous integration, and run prospective studies with human oversight.

Limitations

This study has several limitations. First, the evaluated tasks are standardized proxies (MedMCQA, MedQA, JailbreakBench) rather than live clinical workflows, so the results do not establish clinical efficacy or safety. Second, the PHI protection we implement is deliberately minimal, focusing on coarse contact-identifier masking; richer de-identification and linkage to institutional privacy policies are future work. Third, the adversarial evaluation uses classification-only mode on prompt-level misuse, without exercising long-horizon tool use, system-prompt compromise, or multi-step agent plans, and we do not claim full robustness against future, unseen jailbreak families. Fourth, the safety classifier is instantiated as a prompted LLM rather than a bespoke detector; its behavior depends on the underlying model and few-shot specification, and detecting genuinely novel disallowed behaviors remains an open research problem. Finally, false-positive rates for benign prompts reflect a policy-strict operating point; different deployments may choose more permissive configurations or lighter gates (for example, pattern-only for low-risk intents) based on local risk tolerance, latency budgets, and human oversight capacity.

Ethical Statement

Our experiments use only publicly available benchmarks (MedMCQA, MedQA, JailbreakBench) and do not involve live patient data or personally identifiable information. The iCRAFT runtime is designed to reduce the risk of harmful outputs by enforcing input-side policy and logging all decisions for audit, but it does not replace clinical judgement or institutional oversight. Any real-world deployment in healthcare or other high-stakes domains would require domain-specific governance, regulatory review, and prospective evaluation with human supervision.

References

Blumenthal, D.; and Marellapudi, A. 2025. More Fragmented, More Complex: State Regulation of AI in Health Care.

Booth, H.; Souppaya, M.; Vassilev, A.; Ogata, M.; Stanley, M.; and Scarfone, K. 2024. Secure Software Development Practices for Generative AI and Dual-Use Foundation Models: An SSDF Community Profile. Technical report, National Institute of Standards and Technology. NIST AI 600-1.

Casimiro, M.; Romano, P.; Garlan, D.; Moreno, G. A.; Kang, E.; and Klein, M. 2021. Self-Adaptation for Machine Learning Based Systems.

Chao, P.; Debenedetti, E.; Robey, A.; Andriushchenko, M.; Croce, F.; Sehwag, V.; Dobriban, E.; Flammarion, N.; Pappas, G. J.; Tramer, F.; et al. 2024a. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *Advances in Neural Information Processing Systems*, 37: 55005–55029.

Chao, P.; Debenedetti, E.; Robey, A.; Andriushchenko, M.; Croce, F.; Sehwag, V.; Dobriban, E.; Flammarion, N.; Pappas, G. J.; Tramèr, F.; Hassani, H.; and Wong, E. 2024b. JailbreakBench: An Open Robustness Benchmark for Jailbreaking Large Language Models. In *NeurIPS Datasets and Benchmarks Track*.

Dmonte, A.; Oruche, R.; Zampieri, M.; Calyam, P.; and Augenstein, I. 2024. Claim verification in the age of large language models: A survey. *arXiv preprint arXiv:2408.14317*.

Donakanti, R.; Jain, P.; Kulkarni, S.; and Vaidhyanathan, K. 2024. Reimagining Self-Adaptation in the Age of Large Language Models. In *2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C)*, 171–174.

Durlach, P.; Fournier, R.; Gottlich, J.; Markwell, T.; McManus, J.; Merrill, A.; and Rhew, D. 2024. The AI Maturity Roadmap: a framework for effective and sustainable AI in health care. *NEJM AI Sponsored*.

Fasha, M.; Rub, F. A.; Matar, N.; Sowan, B.; Al Khaldy, M.; and Barham, H. 2024. Mitigating the OWASP top 10 for large language models applications using intelligent agents. In *2024 2nd International Conference on Cyber Resilience (ICCR)*, 1–9. IEEE.

Gangavarapu, A. 2024. Enhancing guardrails for safe and secure healthcare ai. *arXiv preprint arXiv:2409.17190*.

Gaurav, S.; Heikkonen, J.; and Chaudhary, J. 2025. Governance-as-a-Service: A Multi-Agent Framework for AI System Compliance and Policy Enforcement. *arXiv preprint arXiv:2508.18765*.

Hsiung, L.; Pang, T.; Tang, Y.-C.; Song, L.; Ho, T.-Y.; Chen, P.-Y.; and Yang, Y. 2025. Why LLM Safety Guardrails Collapse After Fine-tuning: A Similarity Analysis Between Alignment and Fine-tuning Datasets. *arXiv preprint arXiv:2506.05346*.

Huang, Y.; Gao, C.; Wu, S.; Wang, H.; Wang, X.; Zhou, Y.; Wang, Y.; Ye, J.; Shi, J.; Zhang, Q.; et al. 2025. On the trustworthiness of generative foundation models: Guideline, assessment, and perspective. *arXiv preprint arXiv:2502.14296*.

Inan, H.; Upasani, K.; Chi, J.; Rungta, R.; Iyer, K.; Mao, Y.; Tontchev, M.; Hu, Q.; Fuller, B.; Testuggine, D.; et al. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*.

Jin, D.; Pan, E.; Oufattole, N.; Weng, W.-H.; Fang, H.; and Szolovits, P. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14): 6421.

Kang, M.; and Li, B. 2024. R2-Guard: Robust Reasoning Enabled LLM Guardrail via Knowledge-Enhanced Logical Reasoning. *arXiv preprint arXiv:2407.05557*.

Krawiecka, K.; and de Witt, C. S. 2025. Extending the OWASP Multi-Agentic System Threat Modeling Guide: Insights from Multi-Agent Security Research. *arXiv preprint arXiv:2508.09815*.

Kulkarni, S.; Marda, A.; and Vaidhyanathan, K. 2023. Towards Self-Adaptive Machine Learning-Enabled Systems Through QoS-Aware Model Switching. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 1721–1725.

Li, J.; and Kim, J.-E. 2024. Superficial safety alignment hypothesis. *arXiv preprint arXiv:2410.10862*.

Li, S.; Yao, L.; Zhang, L.; and Li, Y. 2024. Safety layers in aligned large language models: The key to llm security. *arXiv preprint arXiv:2408.17003*.

Liu, Y.; Gao, H.; Zhai, S.; Xia, J.; Wu, T.; Xue, Z.; Chen, Y.; Kawaguchi, K.; Zhang, J.; and Hooi, B. 2025. Guardreasoner: Towards reasoning-based llm safeguards. *arXiv preprint arXiv:2501.18492*.

Marda, A.; Kulkarni, S.; and Vaidhyanathan, K. 2024. SWITCH: An Exemplar for Evaluating Self-Adaptive ML-Enabled Systems. In *Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '24, 143–149. New York, NY, USA: Association for Computing Machinery. ISBN 9798400705854.

NIST AI. 2024. Reducing Risks Posed by Synthetic Content. White paper.

Ong, J. C. L.; Chang, S. Y.-H.; William, W.; Butte, A. J.; Shah, N. H.; Chew, L. S. T.; Liu, N.; Doshi-Velez, F.; Lu, W.; Savulescu, J.; et al. 2024. Ethical and regulatory challenges of large language models in medicine. *The Lancet Digital Health*, 6(6): e428–e432.

Organization, W. H. 2024. *Ethics and governance of artificial intelligence for health: large multi-modal models. WHO guidance*. World Health Organization.

Pal, A.; Umapathi, L. K.; and Sankarasubbu, M. 2022. MedMCQA: A Large-scale Multi-Subject Multi-Choice Dataset for Medical domain Question Answering. In Flores, G.; Chen, G. H.; Pollard, T.; Ho, J. C.; and Naumann, T., eds., *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, 248–260. PMLR.

Rebedea, T.; Dinu, R.; Sreedhar, M.; Parisien, C.; and Cohen, J. 2023. Nemo guardrails: A toolkit for controllable and safe llm applications with programmable rails. *arXiv preprint arXiv:2310.10501*.

Rossi, S.; Michel, A. M.; Mukkamala, R. R.; and Thatcher, J. B. 2024. An early categorization of prompt injection attacks on large language models. *arXiv preprint arXiv:2402.00898*.

Smuha, N. A. 2025. Regulation 2024/1689 of the Eur. Parl. & Council of June 13, 2024 (EU Artificial Intelligence Act). *International Legal Materials*, 1–148.

Tedla, M.; Kulkarni, S.; and Vaidhyanathan, K. 2024. EcoMLS: A Self-Adaptation Approach for Architecting Green ML-Enabled Systems. In *2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C)*, 230–237.

Tirupathi, S.; Salwala, D.; Daly, E.; and Vejsbjerg, I. 2025. GAF-Guard: An Agentic Framework for Risk Management and Governance in Large Language Models. *arXiv preprint arXiv:2507.02986*.

Wang, Z.; Cheng, Z.; Zhu, H.; Fried, D.; and Neubig, G. 2024. What are tools anyway? a survey from the language model perspective. *arXiv preprint arXiv:2403.15452*.

Wiest, I. C.; Ferber, D.; Zhu, J.; van Treeck, M.; Meyer, S. K.; Juglan, R.; Carrero, Z. I.; Paech, D.; Kleesiek, J.;

Ebert, M. P.; et al. 2024. Privacy-preserving large language models for structured medical information retrieval. *NPJ Digital Medicine*, 7(1): 257.

Zeng, W.; Liu, Y.; Mullins, R.; Peran, L.; Fernandez, J.; Harkous, H.; Narasimhan, K.; Proud, D.; Kumar, P.; Radharapu, B.; et al. 2024a. Shieldgemma: Generative ai content moderation based on gemma. *arXiv preprint arXiv:2407.21772*.

Zeng, Y.; Wu, Y.; Zhang, X.; Wang, H.; and Wu, Q. 2024b. Autodefense: Multi-agent llm defense against jailbreak attacks. *arXiv preprint arXiv:2403.04783*.