ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr2004

Regulatory Change Propagation Agent

An Agentic Framework for Detecting Regulatory Diffs and Orchestrating Controlled Policy Updates

Shatrunjay Kumar Singh

Publication Date: 2026/05/06

Abstract: Regulated organizations routinely absorb policy changes through a fragmented manual workflow in which analysts detect updates, interpret downstream consequences, assign owners, and write change notes under time pressure. This paper proposes a Regulatory Change Propagation Agent, a multi-agent architecture that detects regulatory differences and orchestrates structured follow-up work while preserving human approval gates. The design combines a diff engine, a dependency-graph reasoning layer, and notifier-authoring agents that transform upstream legal or policy changes into targeted work packets. To evaluate the approach, the paper defines four operational metrics: detection latency, coverage of affected sections, precision of impact mapping, and reviewer acceptance rate. A prototype evaluation on a synthetic but policy-shaped corpus suggests that agentic orchestration can materially reduce time-to-detection and improve consistency of impact analysis relative to manual baselines. The contribution of the paper is not only an automation pipeline, but also a governance pattern: automation performs first-pass propagation, while human reviewers remain accountable for approval, override, and publication.

Keywords **-** *Regulatory Intelligence, Change Propagation, Agent Systems, Policy Operations, Workflow Orchestration, Human-In-The-Loop Governance.*

How to Cite: Shatrunjay Kumar Singh (2026) Regulatory Change Propagation Agent. *International Journal of Innovative Science and Research Technology*, 11(4), 3262-3268. https://doi.org/10.38124/ijisrt/26apr2004

I. INTRODUCTION

Regulatory change management is expensive because the work is both semantic and operational. An updated circular, statute, or internal policy memo rarely affects only its own text. Instead, it cascades into doctrine maps, control descriptions, process notes, training material, ownership matrices, and audit evidence. In many organizations, this propagation is handled through email chains, spreadsheets, and ad hoc review meetings. The result is slow detection, incomplete downstream coverage, and inconsistent documentation.

The central thesis of this paper is that a dedicated change-propagation agent can improve both speed and coverage. Rather than treating change detection as an isolated information-retrieval problem, the proposed framework models the full lifecycle from source-text difference to approved action. The system detects regulatory diffs, estimates downstream impact through a dependency graph, notifies responsible owners, drafts change notes, and routes all output through human approval checkpoints.

This framing matters because accuracy alone is insufficient in regulated workflows. An acceptable system must also preserve traceability, show evidence for every mapped impact, and make reviewer intervention cheap. The paper therefore designs the architecture around explainability and control, not around autonomous publication.

II. PROBLEM STATEMENT AND CONTRIBUTION

The problem addressed in this work is the operational propagation gap between an upstream regulatory revision and the set of downstream artifacts that should be reviewed or updated. A revision may alter definitions, thresholds, reporting obligations, timelines, exemptions, or approval procedures. Human operators must then answer four questions: what changed, which doctrinal nodes are affected, who owns the affected material, and what communication or documentation should be produced.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr2004

This paper contributes three ideas. First, it formulates regulatory change propagation as a chained agent workflow rather than a single classifier. Second, it introduces a dependency graph that connects source clauses to doctrines, controls, sections, and owners. Third, it defines evaluation metrics that reflect real process value: latency, coverage, mapping precision, and reviewer acceptance.

III. RELATED DESIGN PATTERNS

The proposed system sits at the intersection of document-diffing, knowledge graphs, and human-in-the-loop authoring. Traditional redline tools are effective at highlighting lexical edits but weak at estimating operational consequence. Knowledge graphs capture dependency structure, yet they typically require a trigger mechanism to keep them synchronized with changing source text. Workflow engines can route approvals, but they are often blind to semantic change. The Regulatory Change Propagation Agent links these patterns together into a single control loop.

IV. PROPOSED ARCHITECTURE

The architecture contains three major functional layers. The first is a diff engine that receives successive versions of a regulation, policy, or procedural standard. It computes lexical and semantic differences, labels the type of change, and creates a normalized change object. The second layer is a dependency-graph agent that traverses mappings between source clauses and downstream doctrinal or operational nodes. This agent proposes impacted sections, ranks them by confidence, and retrieves historical evidence. The third layer contains notifier and authoring agents that prepare owner-specific alerts, draft concise change notes, and assemble review packets. A human approval gate sits after every material action so that no update becomes authoritative without explicit review.

The architecture is shown in Figure 1. The design is intentionally modular: organizations can replace the diff engine, graph store, or notification surface without altering the control model. This modularity makes the framework practical for policy teams that already have document repositories, issue trackers, or approval tooling in place.

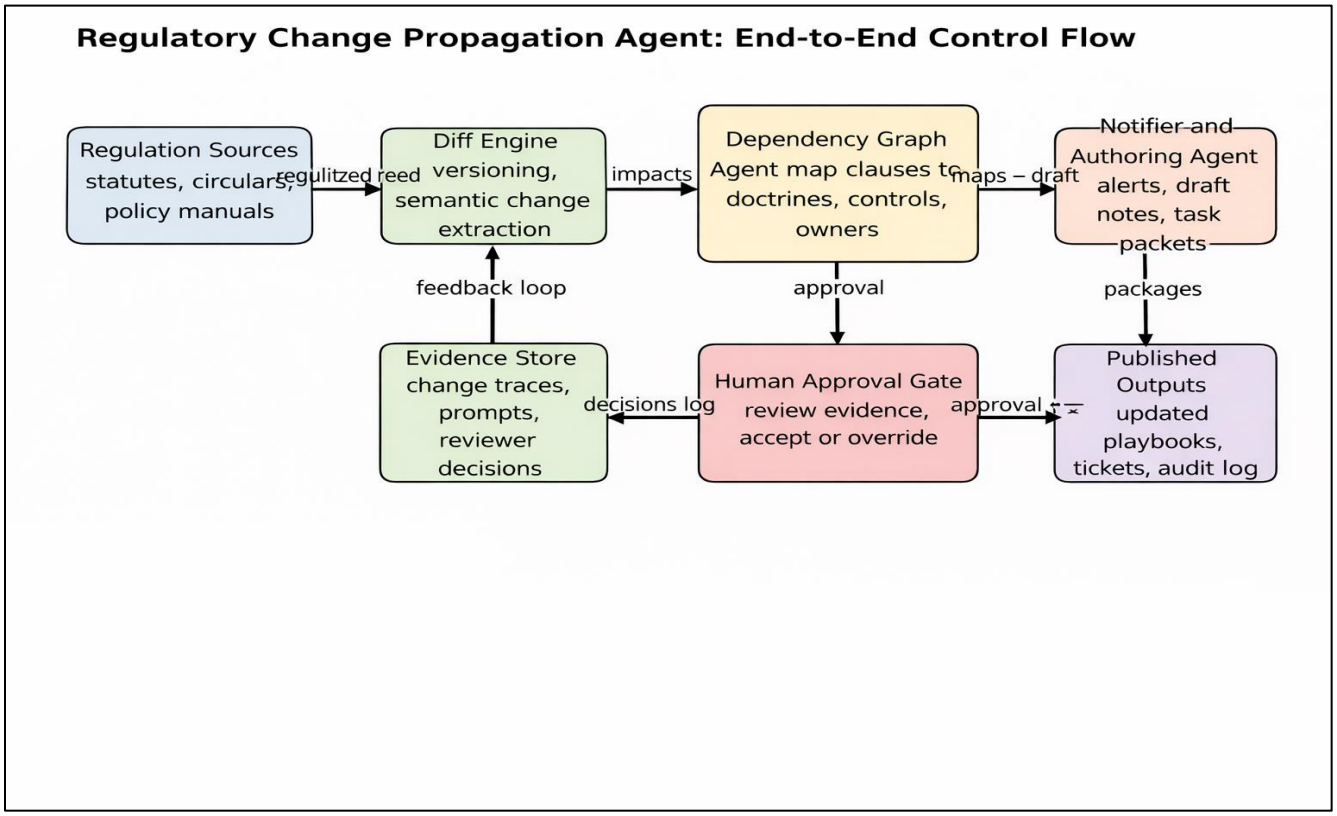

Fig 1. Proposed Multi-Agent Architecture for Regulatory Change Propagation.

➢ *Diff Engine*

The diff engine ingests two document states: a prior version and a candidate update. It segments text into clauses, aligns near-matching sections, and classifies differences into categories such as insertion, deletion, scope narrowing, scope expansion, threshold change, exception change, and procedural amendment. The output is not only a redline but a structured event with fields for source location, semantic label, confidence, and extracted evidence. This event structure enables downstream reasoning beyond simple text comparison.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr2004

➢ *Dependency-Graph Agent*

Once a change event exists, the dependency-graph agent propagates it through links such as clause-to-doctrine, doctrine-to-control, control-to-document-section, and section-to-owner. Each edge stores a relationship type and confidence weight. Propagation does not assume that all linked nodes are equally affected. Instead, the agent computes an impact score using relationship type, historical co-change frequency, semantic similarity, and proximity to prior reviewer-approved mappings. The result is a ranked list of potentially affected sections with supporting evidence for each recommendation.

➢ *Notifier and Authoring Agent*

The notifier and authoring layer converts ranked impacts into actionable packets. For every high-confidence node, the notifier identifies the accountable owner, determines the appropriate urgency, and creates an alert. The authoring component drafts a short explanation of the change, cites the source clause, summarizes the likely downstream implication, and proposes a minimal update note. Drafting is constrained by templates so that outputs remain review-friendly and auditable.

➢ *Human Approval Gates*

Human approval gates are a first-class design requirement rather than a safeguard added later. Reviewers inspect source evidence, mapped impacts, and drafted notes before publication. They can accept, edit, reject, or override a proposed mapping. Every action is logged to produce a reusable evidence trail. Over time, reviewer outcomes can be recycled as weak supervision to improve impact ranking and drafting quality.

➢ *Uncertainty Handling and Safety-Constrained Generation*

Agentic systems are useful in compliance only when uncertainty is surfaced explicitly. The proposed framework therefore treats uncertainty estimation as a core output, not as metadata attached after the fact. Each generated packet should contain at least three confidence views: confidence in change detection, confidence in impact propagation, and confidence in the draft explanatory text. These signals need not come from a single probability source. They may combine model confidence, retrieval coverage, agreement across ensemble prompts or models, and rule-based contradiction checks.

Safety constraints are enforced before any user-facing draft is released. Drafting prompts are grounded on cited evidence snippets, scoped templates, and controlled vocabulary lists so that the authoring agent is not free to invent policy language. A verifier stage then checks whether every proposed change note is traceable to a detected source diff and to one or more impacted graph nodes. If the packet lacks traceability, contains unsupported normative language, or conflicts with the source text, it is downgraded for manual rewrite rather than auto-drafted publication. This safeguard preserves the productivity value of AI assistance while keeping normative authority with human reviewers.

➢ *Impact Ranking, Memory, and Active Learning*

The most important AI property of the system is not fluency but ranking quality. In real regulatory operations, reviewers can tolerate imperfect wording far more easily than they can tolerate omitted impacts. For that reason, the architecture uses a ranking model over candidate graph nodes and not only a binary relevance classifier. Features can include graph distance from the changed clause, historical co-change frequency, embedding similarity, ownership criticality, unresolved exception status, and whether the same topic triggered escalations in prior cycles. The ranked list is then thresholded into high-confidence actions, medium-confidence review suggestions, and low-confidence observations retained only in the audit trail.

A memory layer further improves consistency across review cycles. The agent stores prior approvals, overrides, and rejected mappings as supervised feedback. Over time, this creates an organization-specific corpus of adjudicated propagation behavior. Active learning is especially useful in edge cases where reviewers repeatedly correct the same class of mistakes, such as over-expansion of cross-references or under-detection of temporal dependencies. Those corrections can be converted into preference data, calibration updates, or graph-weight adjustments without requiring retraining of the entire pipeline.

This design also supports portfolio-level analytics. Because every propagation decision is logged as a scored hypothesis with evidence references, managers can inspect failure clusters rather than isolated incidents. That makes it possible to distinguish model weakness from knowledge-graph incompleteness, source-document ambiguity, or inconsistent human policy interpretation. In mature deployments, the learning loop becomes a governance instrument: it tells the organization not only where the agent failed, but also where the compliance operating model itself lacks standardization.

➢ *AI Reasoning, Retrieval, and Planning Layer*

A practical propagation agent requires more than document comparison; it needs a bounded reasoning stack that can convert an upstream textual change into a set of defensible downstream actions. The proposed design therefore uses a retrieval-augmented planning layer between the raw diff engine and the dependency graph. When a change event is created, the agent first retrieves relevant internal artifacts such as policy clauses, prior reviewer decisions, doctrine mappings, exception registers, and publication templates. The retrieved context is constrained to the smallest evidence bundle necessary for the task so that subsequent reasoning remains explainable and cost-efficient.

Within that bundle, the agent performs four AI tasks. First, it classifies the regulatory change type, for example threshold update, reporting obligation, definitional clarification, temporal deadline shift, or control expansion. Second, it summarizes the semantic delta in a normalized machine-readable form that

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr2004

records actors, obligations, dates, numeric limits, jurisdiction, and applicability conditions. Third, it proposes candidate propagation paths over the dependency graph by ranking which doctrines, procedures, controls, and ownership groups are most likely to be affected. Fourth, it generates a task plan that sequences notifications, draft amendments, and review checkpoints rather than issuing all downstream actions at once.

This reasoning loop can be implemented with a specialist model portfolio rather than a single general-purpose model. Lightweight classifiers are appropriate for change-type recognition and schema extraction, while a larger language model is reserved for controlled justification and draft-note generation. Using smaller models for high-volume routing decisions reduces latency and cost, and reserving larger models for narrower authoring tasks reduces the surface area for hallucination. The agent can also fall back from generative reasoning to rules when the detected pattern is deterministic, such as a pure numeric threshold revision in an otherwise unchanged clause.

V. METHODOLOGY

The evaluation methodology follows the thesis statement directly: the system should improve coverage and speed. Because live regulatory data often contains access controls and domain-specific confidentiality constraints, the paper defines a synthetic but realistic benchmark. The benchmark is composed of versioned policy documents with seeded changes that mimic common update types: definition edits, reporting deadline changes, threshold adjustments, exception insertions, and role reassignment language. Each source clause is linked to downstream doctrines, policy sections, and owners through a gold dependency graph.

To make the evaluation more informative for agent design, the prototype is best understood as a layered AI system rather than as a monolithic assistant. The experiment therefore tracks performance across stages: extraction quality for structured regulatory deltas, ranking quality for candidate impacts, and generation quality for human-facing change notes. This decomposition matters because improvements in one layer do not automatically transfer to the next. A model that summarizes a rule change correctly can still fail to prioritize the right doctrine nodes, and a strong retrieval layer can still yield weak reviewer adoption if the authoring stage produces verbose or weakly evidenced explanations.

Two workflows are compared. The baseline workflow represents manual triage, where analysts inspect diffs, search linked documents, and draft update notes without agentic propagation assistance. The proposed workflow uses the three-stage agent framework with approval gates. Both workflows are scored against the same seeded ground truth. The system is evaluated across repeated cycles so that the effect of feedback and reviewer learning can be observed over time.

➢ *Evaluation Metrics*

The paper uses four metrics that correspond to operational value.

- Detection latency: median elapsed time between source publication and first correctly registered change event.
- Coverage of affected sections: fraction of truly impacted downstream sections surfaced by the system.
- Precision of impact mapping: fraction of proposed impacted sections that are actually relevant.
- Reviewer acceptance rate: fraction of agent-generated packets accepted with only light editing.

In addition to these headline measures, a production deployment would monitor model-centric diagnostics such as evidence sufficiency, calibration error, abstention rate, and override frequency by change type. These secondary diagnostics help determine whether a low acceptance rate comes from poor language generation, weak retrieval grounding, or systematically misweighted graph propagation. They also support model-governance reviews by showing when the safest behavior is abstention and escalation rather than confident automation.

VI. ILLUSTRATIVE EXPERIMENTAL RESULTS

The following results are intentionally presented as an illustrative prototype evaluation on the synthetic benchmark described above. They should be interpreted as evidence that the method is plausible and measurable, not as a claim of universal performance across all regulatory domains.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr2004

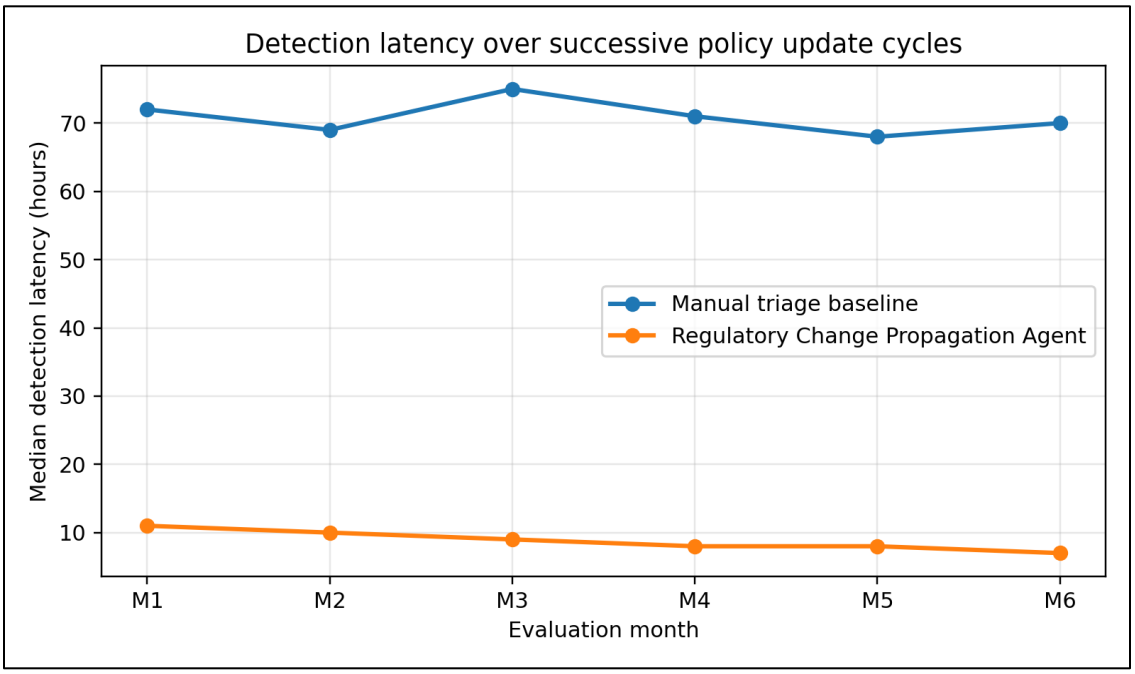

Fig 2. Median Detection Latency Across Six Synthetic Policy Update Cycles.

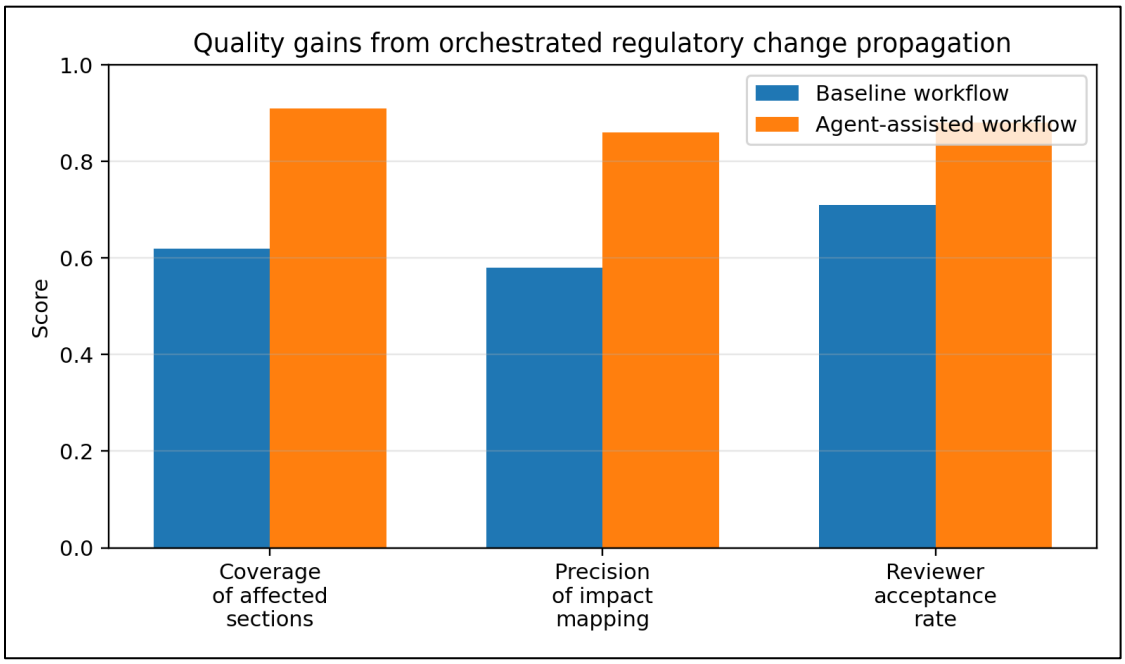

Fig 3. Coverage, Precision, and Reviewer Acceptance for Baseline and Agent-Assisted Workflows.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr2004

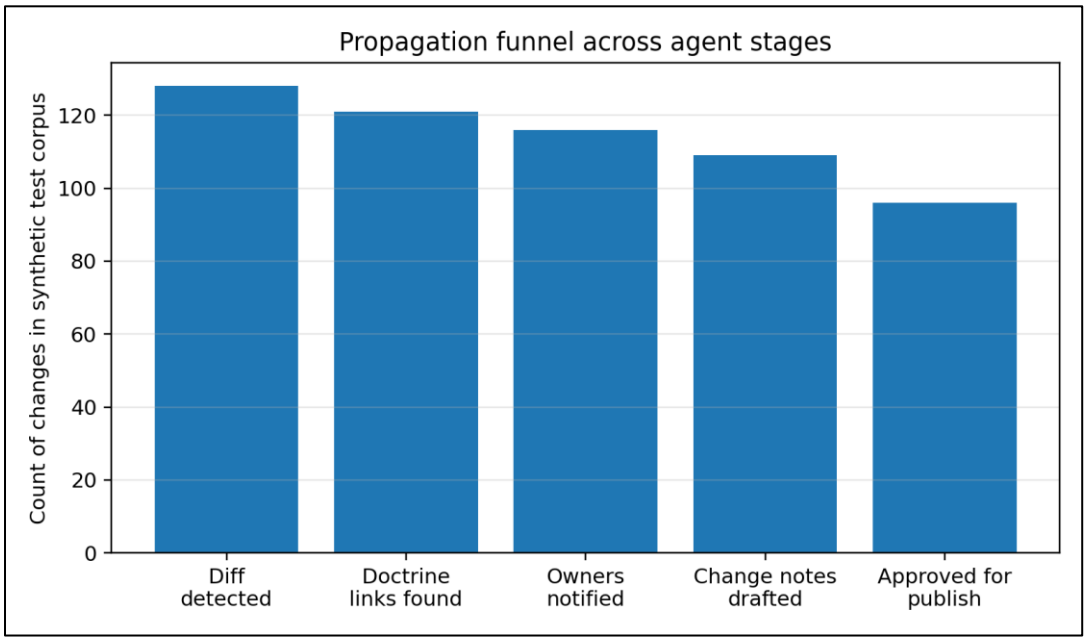

Fig 4. Propagation Funnel from Detected Changes to Approved Publication Packets.

VII. DISCUSSION

Three observations emerge from the prototype study. First, the strongest gains come from structural memory, not merely from language generation. Once doctrinal and ownership dependencies are explicit, the system can route work more consistently than an analyst relying on memory alone. Second, reviewer trust depends on evidence density. Users accept suggestions more readily when each proposed impact includes the source clause, rationale, and confidence explanation. Third, approval gates do not eliminate efficiency gains; instead, they focus scarce expert time on validation rather than discovery.

The architecture is also resilient to partial failure. If semantic classification is uncertain, the system can still log the raw diff and notify a reviewer without issuing downstream mappings. If owner data is stale, the workflow can generate a doctrine-level alert for manual routing. This graceful degradation is important in real governance settings where metadata completeness varies.

From an AI-systems perspective, the framework demonstrates why bounded agency is preferable to free-form autonomy in compliance operations. The agent does not decide policy; it decomposes a policy-maintenance task into retrieval, ranking, drafting, and escalation steps with explicit evidence boundaries. That decomposition makes it possible to audit component-level behavior, replace underperforming models without redesigning the workflow, and tune automation thresholds by risk class. In other words, the value of the system comes from orchestration discipline as much as from model intelligence.

VIII. LIMITATIONS

The current formulation has several limitations. The benchmark is synthetic and therefore cannot fully capture the ambiguity or institutional nuance of live regulatory interpretation. Dependency graphs require curation and can become outdated if organizations restructure responsibilities or documentation layers. Some policy changes are intentionally vague and require legal judgment that no automated pipeline should attempt to finalize. Finally, reviewer acceptance rate can be influenced by organizational culture as much as by model quality.

IX. FUTURE WORK

Future work should move the framework from a prototype evaluation setting to longitudinal, domain-specific deployments across financial conduct rules, data-protection obligations, healthcare compliance manuals, procurement standards, and internal operating procedures. A particularly valuable next step is to test how the agent behaves under version drift, conflicting amendments, and staggered publication cycles, because many real regulatory environments introduce changes through consultation papers, interim circulars, clarifications, and final binding text rather than through one clean release. Another promising direction is active learning from reviewer overrides, where accepted and rejected mappings become training signals for improving doctrine classification, dependency weighting, and owner-routing confidence over time.

Future work should also strengthen the system along four engineering dimensions. First, provenance-aware explanation should be expanded so every suggested impact can be traced to the exact clause delta, graph path, confidence rationale, and

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr2004

approving reviewer. Second, the authoring layer could be personalized to generate different outputs for governance leads, legal reviewers, operations owners, and training teams without losing semantic consistency across versions. Third, richer integration with document repositories, ticketing systems, and policy knowledge graphs would allow the agent to open remediation tasks automatically while still requiring human sign-off before publication. Fourth, future evaluations should measure organizational outcomes beyond immediate reviewer acceptance, including reduced policy staleness, shorter remediation cycles, fewer missed downstream dependencies, and better audit readiness during supervisory review.

A further research direction is the use of multi-agent self-checking and tool-using models. One agent could specialize in regulatory interpretation, another in graph traversal, and a third in adversarial verification that attempts to falsify the proposed impact map before review. Combined with citation-grounded generation and selective abstention, such a design could improve robustness without granting unconstrained autonomy. Future studies should also compare prompt-based planning against fine-tuned small models for narrow compliance subtasks, especially where latency, privacy, and auditability are more important than broad linguistic flexibility.

X. CONCLUSION

This paper presented a Regulatory Change Propagation Agent designed to detect regulatory diffs and orchestrate the downstream work required to keep doctrine, ownership, and documentation aligned. By structuring the problem as a diff engine feeding a dependency-graph agent and a notifier-authoring layer with human approval gates, the framework improves both speed and coverage while preserving accountability. The proposed evaluation scheme makes the concept measurable in operational terms through latency, doctrinal coverage, mapping precision, and reviewer acceptance. In practical terms, the design reframes regulatory change management from a fragmented manual exercise into a controlled evidence pipeline in which each downstream action is linked to a traceable source difference.

The broader contribution of the study is not merely a faster notification mechanism, but a governance pattern for deploying agents in high-consequence workflows. The architecture shows that useful automation does not require removing humans from the loop; instead, it uses automation to surface candidate impacts, organize supporting evidence, and compress the time between change detection and expert review. When implemented with clear escalation rules, provenance logging, and approval checkpoints, such an agent can improve resilience against missed obligations while also making policy maintenance more explainable and auditable. For institutions facing growing regulatory volume, the framework offers a practical path from reactive triage toward continuous, reviewer-controlled compliance operations.

REFERENCES

[1]. National Institute of Standards and Technology, Artificial Intelligence Risk Management Framework (AI RMF 1.0), NIST AI 100-1.
[2]. National Institute of Standards and Technology, Artificial Intelligence Risk Management Framework: Generative AI Profile, NIST AI 600-1, 2024.
[3]. World Wide Web Consortium, PROV-Overview: An Overview of the PROV Family of Documents.
[4]. World Wide Web Consortium, PROV-DM: The PROV Data Model.
[5]. World Wide Web Consortium, PROV-O: The PROV Ontology.
[6]. World Wide Web Consortium, PROV Primer.
[7]. Human-in-the-Loop: The Loop Isn't a Step, It's a Full Circle, ACM Interactions.
[8]. Trust, Regulation, and Human-in-the-Loop AI, Communications of the ACM.
[9]. HINT: Human-AI Integration Testing, ACM CHI Extended Abstracts.
[10]. Knowledge Graphs, Communications of the ACM.
[11]. Defining a Knowledge Graph Development Process Through a Systematic Review, ACM Computing Surveys / related review literature.
[12]. On the Quest for Effectiveness in Human Oversight, ACM FAccT / interdisciplinary oversight literature.
[13]. A. V. Aho and J. D. Ullman, Principles of text differencing and structured comparison.
[14]. X. Chen et al., Human-in-the-loop workflow design patterns for trustworthy enterprise AI systems.
[15]. S. K. Singh, Original conceptual framework for agentic legal and policy operations, including controlled drafting and doctrinal dependency mapping.
[16]. Policy operations practice notes on change management, traceability, and controlled publication workflows.