Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

A Unified Framework for Self-Evolving and Self-Healing Artificial Intelligence Agents

Prudvi Saisaran Ponduru

Researcher, Department of Computer Science

Abstract:
Artificial intelligence agents are increasingly expected to operate for long periods, use external tools, adapt to changing environments and recover from failures without continuous human supervision. Current work, however, is divided between self-evolving agents that improve through memory, reflection, continual learning, reinforcement learning or evolutionary search, and self-healing systems that monitor, diagnose, contain, repair and verify operational faults. This paper integrates these traditions into a unified Self Evolving and Self-Healing Agent framework. A four-axis taxonomy is introduced across the object of change, timescale of adaptation, recovery boundary and assurance mechanism. A dual loop architecture is then proposed in which a fast runtime healing loop handles detection, diagnosis, containment, repair and verification, while slower evidence gated evolution loop updates durable artifacts such as structured memory, prompts, workflows, tool skills, code patches and model adapters. Formal objectives are specified for utility, risk, recovery cost, retention and transfer, and a selective persistence rule is defined to prevent emergency behavior from becoming an unsafe permanent update. A multi domain, fault injected evaluation protocol is presented with baselines, ablations, benchmarks and metrics for task success, adaptation gain, retention, transfer, detection delay, time to recovery, degradation area, false recovery and safety violations. The analysis shows that self-evolution and self-healing should be evaluated as complementary capabilities under runtime assurance, auditability, interpretability and staged governance. The paper is a conceptual and methodological contribution; it does not claim new benchmark results for the proposed architecture.

Keywords: Agentic Artificial Intelligence, Self-Evolution, Self-Healing, Continual Learning, Runtime Assurance, Autonomous Recovery.

1. Introduction

Artificial intelligence agents differ from one shot language models because they pursue goals through sequences of reasoning, tool use and environment interaction. Modern agent architectures combine planning, action, observation and memory, as illustrated by ReAct, Tree of Thoughts, Reflexion and Voyager [5, 6, 7, 8]. This shift from answer generation to sustained action increases usefulness but also creates a reliability problem: an agent may encounter distribution shift, stale memory, tool outages,

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

interface changes, ambiguous feedback, adversarial instructions or failures caused by its own earlier actions.

Two research traditions address different parts of this problem. The first studies persistent improvement. Self-evolving agents can accumulate skills, distill reasoning traces, generate curricula, refine policies or alter learning procedures across episodes [8, 14, 15, 16, 17, 18, 19, 20]. The second studies dependable operation. Autonomic computing, self-adaptive software, runtime monitoring, automated program repair and formal verification provide mechanisms for detecting, containing and repairing faults [1, 21, 22, 27, 28, 29, 30, 34]. These traditions are increasingly used together, yet they are rarely evaluated under a common model.

The separation is consequential. An adaptive agent without runtime assurance may learn from corrupted feedback, memorize unsafe workarounds or propagate a local emergency response into later tasks. A healing agent without persistent learning may repeatedly rediscover the same repair, incur high recovery cost and remain brittle to novel faults. A unified system must therefore answer two distinct questions at every failure: what must be changed now to restore safe operation, and what evidence is sufficient to justify a durable change for future operation.

Existing benchmarks also leave an integration gap. WebArena, WorkArena, OSWorld, AgentBench, SafeAgentBench, Agent SafetyBench, SWE bench and LifelongAgentBench test important capabilities, but no widely adopted benchmark combines persistent multi episode learning, explicit fault injection, verifiable recovery and governed promotion of agent updates [9, 10, 11, 12, 18, 31, 32, 33]. The absence of such evaluation allows simple retry loops, open book retrieval and genuine self healing to be described with similar language even though their reliability properties differ.

The principal contributions of this paper are as follows:
1. Operational definitions are established for self adaptation, self evolution, self repair and self healing, with explicit distinctions between retrieval based improvement and internalized capability change.
2. A four axis taxonomy is developed to classify what is changed, when change occurs, where recovery is applied and how the result is assured.
3. A dual loop architecture is proposed that separates conservative runtime healing from slower evidence gated evolution of durable artifacts.
4. A formal objective and promotion rule are specified to balance task utility, safety risk, recovery cost, transfer, retention and catastrophic forgetting.
5. A reproducible multi domain evaluation methodology is provided with fault libraries, baselines, ablations and execution based metrics.
6. Safety, security, ethics and governance requirements are integrated as architectural constraints rather than treated as post deployment additions.

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

The remainder of the paper reviews the evidence base, defines the problem, presents the taxonomy and architecture, formalizes the proposed control logic, specifies an evaluation program and examines safety, governance and open research challenges.

2. Structured Evidence Synthesis and Related Work

2.1. Scope and Evidence Selection

A structured narrative synthesis was conducted across peer reviewed conference papers, journal articles, preprints, benchmark papers, technical reports and governance documents. The evidence was organized around five themes: lifelong and experience driven agent improvement, benchmarks for persistent learning, automated repair, runtime assurance and autonomic system design. Priority was given to primary sources that define architectures, algorithms, benchmarks or governance requirements. Because the field is moving rapidly and terminology remains inconsistent, the synthesis is intended to expose design relationships and evaluation gaps rather than to claim an exhaustive systematic review.

2.2. Persistent Agent Improvement

Voyager demonstrated open ended embodied learning through an automatically expanding skill library, while Reflexion used verbal feedback stored in episodic memory to improve subsequent decisions [7, 8]. Later work extended this direction through evaluator guided refinement, synthetic self improvement data, online curriculum reinforcement learning, iterative self training and structured reasoning memory [13, 14, 15, 16, 17]. These methods establish that agents can improve across episodes, but the updated artifact varies. Some systems modify only the inference context, some update a retrieval store, and others fine tune parameters or optimize workflows.

LifelongAgentBench and SE Bench sharpen the distinction between apparent improvement and durable internalization [18, 19]. Raw replay can introduce irrelevant information, context limits and false confidence. Retrieval based agents may perform well when prior traces are available but fail when memory access is constrained. Persistent self evolution should therefore be measured not only by post adaptation task success, but also by retention, transfer and robustness when explicit retrieval is restricted.

2.3. Self Repair and Self Healing

Automated program repair provides the clearest operational examples of agent driven healing. RepairAgent treats bug fixing as an autonomous loop over repository exploration, test execution and code editing, and reported 164 repairs on Defects4J [21]. TraceRepair adds runtime execution traces and multi agent debate to constrain candidate patches [22]. ReVeal emphasizes reliable self-verification during coding agent evolution [23]. These systems show that concrete artifacts can be repaired, but correctness remains bounded by tests, execution coverage and the quality of verification.

The broader self-healing literature originates in autonomic computing. The Monitor Analyze Plan Execute over a Knowledge base pattern defines a closed loop in which telemetry is converted into diagnosis,

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

adaptation plans and controlled execution [1]. Chaos engineering has been proposed as an evaluation method for self-adaptive and self-healing systems because resilience claims must be tested under controlled perturbation rather than inferred from nominal behavior [34]. Modern agent systems can reuse these principles while extending the knowledge base to include prompts, reasoning traces, tool descriptions, health models and safety policies.

2.4. Runtime Assurance and Verification

GuardAgent and ShieldAgent supervise action trajectories through guard reasoning and verifiable policy interpretation [25, 26]. AgentSpec provides executable runtime constraints, Pro2Guard anticipates unsafe states through probabilistic model checking, Agent C enforces temporal constraints during generation, and VeriGuard combines offline policy synthesis with lightweight online monitoring [27, 28, 29, 30]. Reported results indicate that strong policy conformance can be achieved in structured domains, but the cost of formalization and the coverage of specifications remain limiting factors.

AIOS contributes an operating system perspective in which agents receive scheduling, memory, tool and isolation services [24]. This perspective is important because many failures attributed to model reasoning are actually compound failures involving tools, credentials, resource limits, stale interfaces or shared state. Self-healing must therefore operate across semantic, software, infrastructure and socio technical boundaries.

Table 1: Representative Systems Across Evolution and Healing

System	Primary Mechanism	Durable Update	Healing Capability	Main Limitation
Voyager 2023	Skill library and automatic curriculum	Skills and memory	Limited	No explicit runtime assurance [8]
Reflexion 2023	Verbal reinforcement and episodic reflection	Reasoning context	Local correction	May loop or rationalize errors [7]
WebRL 2025	Self generated curriculum reinforcement learning	Policy and curriculum	None	Depends on reward quality [15]
Agent R 2025	Iterative self-training for reflection	Error correction policy	Trajectory repair	No verified safety gate [16]

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

System	Primary Mechanism	Durable Update	Healing Capability	Main Limitation
ReasoningBank 2025	Distilled reasoning memory	Structured memory	Indirect	Retrieval can mask weak internalization [17]
LifelongAgentBench 2025	Multi episode lifelong evaluation	Evaluation only	None	Does not provide a healing mechanism [18]
RepairAgent 2025	Autonomous program repair	Code patches	Software healing	Test bounded correctness [21]
AgentSpec 2025	Executable runtime constraints	No durable learning	Action blocking	Mostly reactive [27]
Pro2Guard 2025	Probabilistic foresight	No durable learning	Proactive intervention	Requires calibrated abstractions [28]
Agent C 2025	SMT constrained temporal safety	No durable learning	Verified action control	Formalization burden [29]
VeriGuard 2025	Verified policy plus online monitor	Verified policies	Runtime assurance	Policy synthesis cost [30]
SE Bench 2026	Knowledge internalization evaluation	Evaluation only	None	Focuses evolution, not healing [19]

2.5. Research Gap

The literature supports strong components but not a complete theory. Persistent learning papers usually treat failure as a source of feedback, while runtime assurance papers usually assume a fixed policy. Repair papers produce candidate fixes but often lack a theory for when a fix should be internalized across tasks. The resulting gap is an architecture and evaluation framework that treats self-evolution and self healing as interacting, governed capabilities.

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

3. Definitions, Research Questions and Problem Formulation

3.1. Operational Definitions

Table 2: Operational Definitions

Concept	Operational Meaning	Typical Artifacts
Self-Adaptive Agent	Changes behavior online in response to feedback or non-stationarity; persistence beyond the current deployment window is not required.	Plan, tool choice or reasoning path
Self-Evolving Agent	Makes persistent changes that improve future episodes and survive beyond immediate context.	Memory, prompts, workflows, tools, parameters or update rules
Self-Repairing Agent	Produces a concrete correction after a fault.	Patch, corrected plan, repaired prompt or safe tool sequence
Self-Healing Agent	Detects, diagnoses, contains, repairs, verifies and restores acceptable operation under bounded risk.	Health state, recovery policy, rollback point and verified repair

The distinction between retrieval and internalization is central. Retrieval based improvement stores evidence outside the agent policy and reuses it when accessible. Internalization changes a durable artifact so that the capability remains available under reduced retrieval support. Both mechanisms are useful, but only the second provides strong evidence of self-evolution [18, 19].

3.2. Research Questions and Hypotheses

Table 3: Research Questions and Testable Hypotheses

ID	Research Question	Hypothesis
RQ1	Can agents improve persistently across long task streams?	Structured memory plus selective internalization will improve transfer and retention over stateless or retrieval only baselines.
RQ2	Do self generated curricula improve adaptation?	Evaluator guided or verifiable curriculum learning will outperform imitation only adaptation after distribution shift.

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

ID	Research Question	Hypothesis
RQ3	Can runtime assurance reduce harmful actions without unacceptable utility loss?	Hybrid rules and probabilistic foresight will reduce violations while retaining more task utility than reactive filtering alone.
RQ4	Can diagnosis aware healing outperform naive retry?	Rollback, replanning, tool substitution, patching and verification will reduce recovery time and false recovery.
RQ5	Does verified recovery improve trustworthiness?	Formal or execution based gates will reduce unsafe patch acceptance and hidden regression.
RQ6	Are evolution and healing complementary?	The unified agent will outperform single capability systems under repeated shifts and injected faults.

3.3. Agent and Environment Model

An agent is modeled as a compound system A = (P, M, T, G, D, R, V, U, S). P is the planner or policy, M is persistent memory, T is the tool and execution layer, G is the runtime monitor, D is the diagnoser, R is the recovery policy, V is the verifier, U is the evolution manager and S is the safety policy. At time t, the system receives observation o_t, selects action a_t, receives result y_t and updates its operational state x_t. Durable changes are represented by delta values applied to one or more artifacts only after verification.

The environment may be partially observable, non-stationary and adversarial. Faults may arise from the model, memory, tool interface, software dependency, resource layer, data source, external actor or human instruction. A health state is therefore multi-dimensional and cannot be inferred from a single confidence score.

4. Four Axis Taxonomy of Evolution and Healing

The proposed taxonomy classifies systems along four axes. The object of change specifies what is modified. The timescale specifies when the modification occurs. The recovery boundary specifies which layer is repaired. The assurance mechanism specifies how the modification is validated. This organization avoids grouping fundamentally different systems under a single self-improving label.

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

Table 4: Four Axis Taxonomy

Axis	Representative Values	Design Consequence
Object of Change	Reasoning trace; episodic memory; semantic memory; prompt; workflow; tool skill; source code; model parameters; learning rule	Determines persistence and blast radius
Timescale	Within step; within episode; inter episode; continual deployment; offline evolutionary search	Determines latency, data volume and verification budget
Recovery Boundary	Semantic; software; infrastructure; security; organizational or human approval	Determines authority and escalation path
Assurance Mechanism	Heuristic check; test suite; runtime rule; probabilistic prediction; formal verification; human review	Determines confidence and cost of promotion

Self-evolution is strongest when durable artifacts are changed and retained across episodes. Self-healing is strongest when faults are explicitly detected, diagnosis is represented, containment is applied, candidate repairs are verified and service is restored within a measurable operating envelope. Systems that merely retry or append a reflection should be described as adaptive recovery unless they demonstrate these properties.

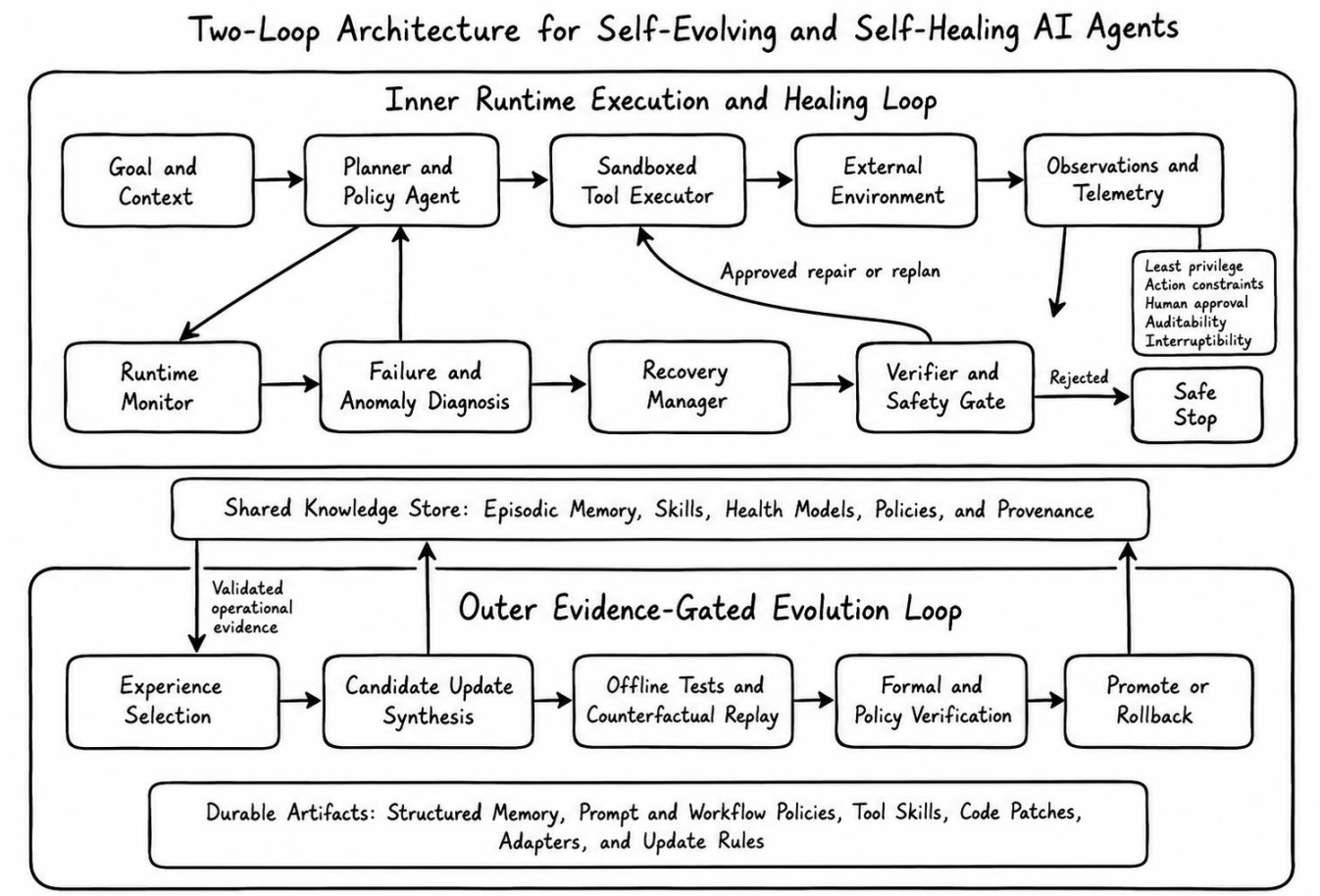
The taxonomy also exposes correlated risk. Updating a prompt affects one interaction pattern, whereas updating shared model weights, a global tool policy or a multi agent memory can alter many future trajectories. Verification requirements should increase with persistence, scope and irreversibility.

5. Proposed Dual Loop Architecture

A dual loop architecture is proposed to separate immediate resilience from durable learning. The inner loop is optimized for low latency, conservative intervention and interruptibility. The outer loop is optimized for evidence quality, reproducibility and controlled promotion. Both loops share a knowledge store, but they have different permissions and decision thresholds.

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

Figure 1: Two-Loop Architecture for Self-Evolving and Self-Healing AI Agents



5.1. Inner Runtime Execution and Healing Loop

The inner loop begins with goal interpretation, planning and sandboxed execution. Telemetry is collected from model outputs, tool calls, external systems, resource usage and safety policy events. When health remains within the accepted envelope, execution continues. When an anomaly is detected, the diagnoser estimates fault class, severity, reversibility and uncertainty. The recovery manager then chooses among selective retry, replanning, tool substitution, checkpoint restoration, code repair, graceful degradation or safe stop.

The verifier gates every recovery that can affect external state. Verification may combine deterministic policy rules, type and schema checks, execution tests, hidden regression tests, simulation, probabilistic model checking and human approval. Rejected candidates are not executed. If no safe recovery exists, the system rolls back, reduces privileges or stops.

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

5.2. Outer Evidence Gated Evolution Loop

The outer loop selects experiences that are informative, repeated, high impact or representative of distribution shift. Candidate updates may include memory consolidation, prompt revision, workflow reordering, tool acquisition, code patching, adapter training or modification of an update rule. Each candidate is evaluated offline through replay, counterfactual tasks, held out tests and safety checks. Only updates that improve validated objectives without new policy violations are promoted.

This separation implements selective persistence. Runtime repair may be intentionally temporary, while evolution requires stronger evidence. A workaround used during an outage, for example, should not become the default tool policy unless it performs safely across representative conditions. This principle reduces the risk that emergency behavior becomes institutionalized.

5.3. Shared Knowledge and Provenance

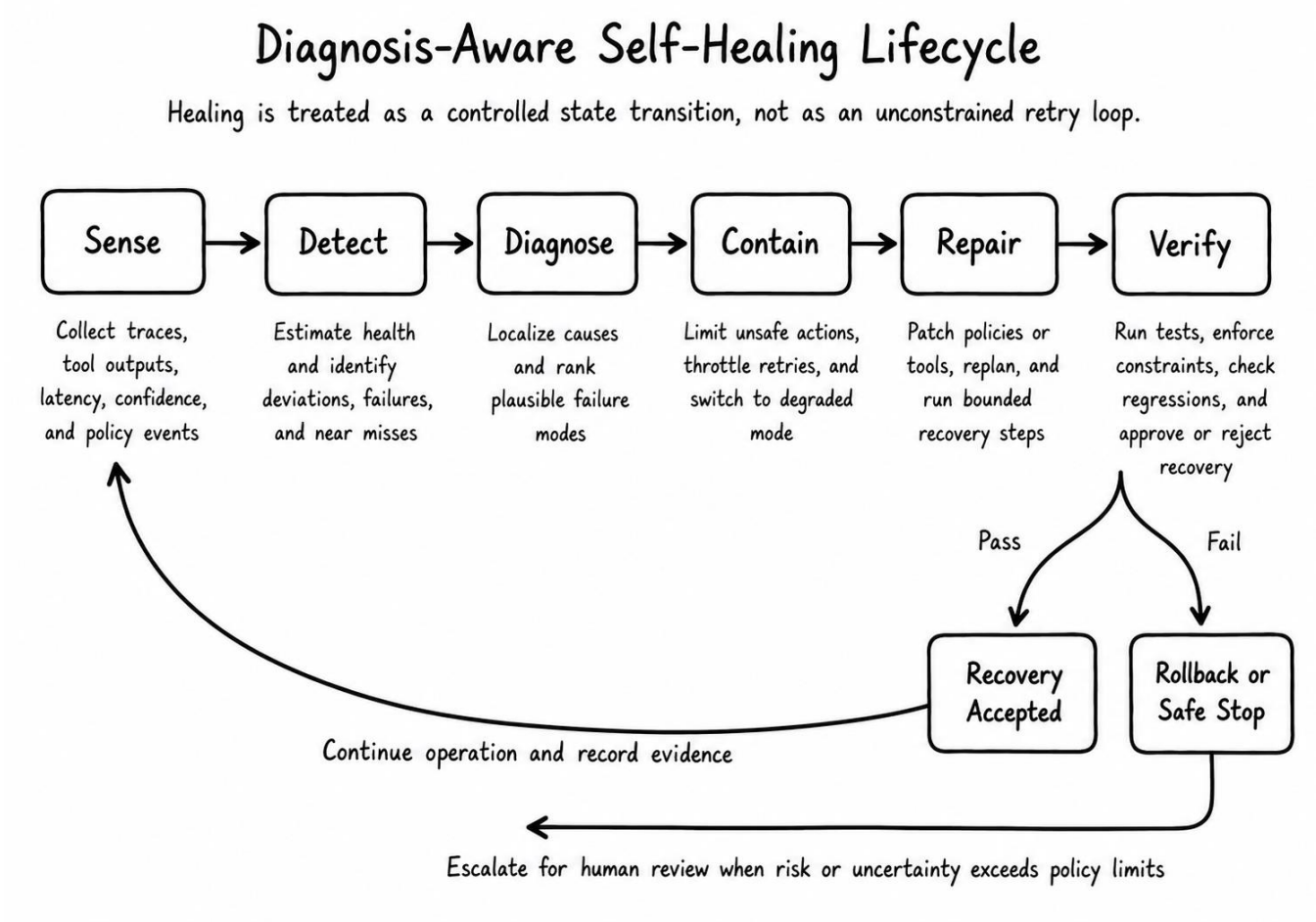
The shared knowledge store contains episodic traces, structured reasoning memories, reusable skills, health models, fault taxonomies, recovery outcomes, safety policies and provenance. Every durable update should record its source episodes, evaluator version, test environment, confidence, approver and rollback target. Provenance supports audit, reproduction and incident analysis and helps distinguish inherited knowledge from newly internalized capability.

5.4. Defense in Depth Controls

Least privilege, action constraints, sandboxing, approval requirements, rate limits, immutable logs and interruptibility form a safety envelope around both loops [24, 35, 36]. The evolution manager should not directly grant itself credentials, expand its action space or weaken verification thresholds. Changes to such controls require a separate governance authority.

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

Figure 2: Diagnosis-Aware Self-Healing Lifecycle



6. Formal Model and Decision Rules

6.1. Health Estimation

Let S_task measure progress toward the goal, S_tool measure tool and interface integrity, S_policy measure policy conformance, and S_resource measure computational and operational health. A composite health score is defined as a weighted value between zero and one.

$$H_t = w_1 S_{task,t} + w_2 S_{tool,t} + w_3 S_{policy,t} + w_4 S_{resource,t}, \quad \text{where} \quad \sum(w_i) = 1 \qquad (1)$$

An anomaly is declared when health falls below a threshold or when the current telemetry distribution departs substantially from a reference distribution. The second condition captures failures that remain superficially successful but exhibit unusual action patterns, latency or policy events.

$$A_t = 1 \text{ if } H_t < \tau_h \text{ or } D(z_t, z_{ref}) > \tau_d; \text{ otherwise } A_t = 0 \qquad (2)$$

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

6.2. Recovery Selection

For a diagnosed fault f_t, the recovery manager generates a candidate set R(f_t). The selected recovery maximizes expected utility while penalizing safety risk and operational cost, subject to hard safety constraints.

r* = arg max over r in R(f_t) of E[U(x_t+1 | r) - lambda Risk(r) - mu Cost(r)] subject to C_safe(r) = 1 (3)

A naive retry is one candidate, not the default policy. Replanning is preferred when the plan is invalid, tool substitution when an interface is degraded, rollback when state corruption is likely, and safe stop when the expected risk cannot be bounded.

6.3. Evolution Objective and Promotion Gate

The outer loop optimizes long term utility while rewarding transfer and penalizing forgetting. Let u_t be task utility, q_t be safety risk, c_t be cost, G_transfer be improvement on unseen related tasks and F_forgetting be degradation on previously learned tasks.

J = E[sum from t = 0 to T of gamma^t (u_t - lambda q_t - mu c_t)] + eta G_transfer - kappa F_forgetting (4)

A candidate durable update delta is promoted only when it exceeds a minimum held out improvement, introduces no new critical safety violation and reaches a required confidence level. Additional conditions may include statistical significance, resource limits and human approval.

Promote(delta) = 1 iff Delta J_holdout >= epsilon, Violations_critical = 0, and Confidence >= alpha (5)

6.4. Resilience Metrics

Time to recovery is measured from the first observable fault to return within the accepted operating envelope. Performance degradation area integrates the shortfall between nominal and observed utility during the fault. False recovery rate measures the fraction of accepted recoveries that later fail hidden tests, violate policy or cause regression. These measures distinguish systems that recover quickly but incorrectly from systems that restore reliable service.

Degradation Area = integral from t_fault to t_recovered of max(0, U_nominal(t) - U_observed(t)) dt (6)

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

7. Unified Self Evolving and Self Healing Algorithm

The following pseudocode places the verifier before external action and before durable learning. This ordering is deliberate. A harmful one time action can cause immediate damage, while a harmful durable update can affect many future tasks and agents.

Algorithm 1: Unified Self Evolving and Self Healing Agent

```
Algorithm Unified Self Evolving and Self Healing Agent
Inputs: task stream T, planner P, memory M, monitor G, diagnoser D,
      recovery policy R, verifier V, evolution manager U and safety policy S

for each task t in T:
   state <- initialize(t); trace <- empty
   while state is not terminal:
      observation <- observe(state)
      context <- M.retrieve(observation, trace)
      action <- P.propose(observation, context, S)
      if V.pre_action_check(action, S) is false:
         action <- R.safe_alternative(observation, context, S)
      result <- execute_in_sandbox(action)
      append (observation, action, result) to trace
      health <- G.assess(trace, result)
      if health indicates anomaly or failure:
         fault <- D.classify(trace, result, health)
         candidates <- R.generate_repairs(fault, trace, context)
         approved <- first candidate c for which V.validate(c, fault, S) is true
         if approved exists: state <- apply(approved, state)
         else: state <- rollback_or_safe_stop(state)
      else: state <- transition(state, result)
   outcome <- evaluate_task(trace); M.update(outcome, trace)
   if U.should_propose_update(outcome, trace):
      delta <- U.synthesize_update(M, trace, S)
      if V.posthoc_validate(delta, heldout_tests, S): U.commit(delta)
```

The algorithm can be implemented with a single model or a multi agent organization. Separation of roles is recommended in high risk domains because a planner that proposes a repair should not be the sole authority that approves it. Independent evaluators, deterministic contracts and hidden tests reduce self confirmation bias.

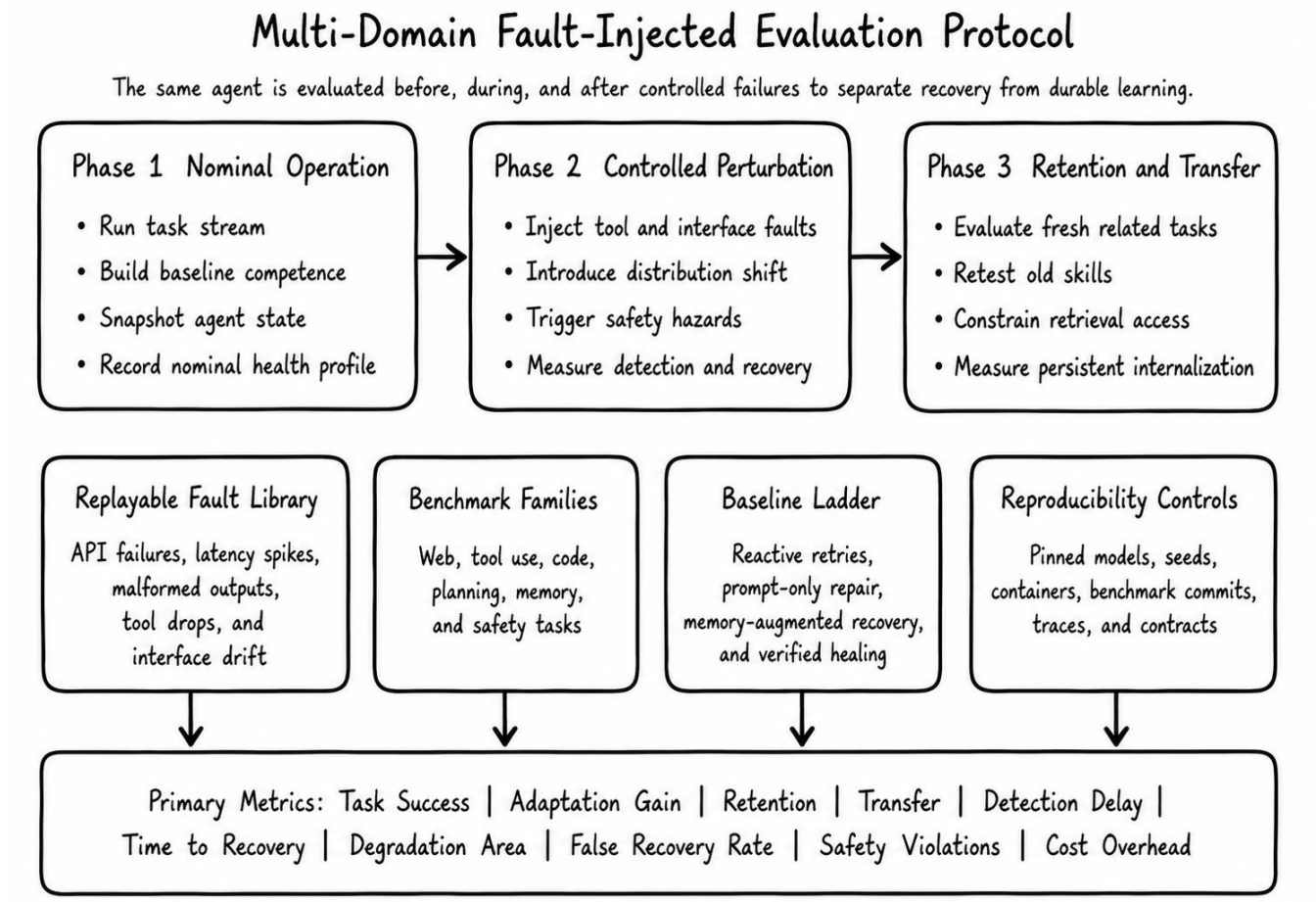
Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

8. Experimental Methodology

8.1. Three Phase Protocol

A credible evaluation must be multi domain, fault injected and execution based. The same agent should be measured in three phases. Nominal operation establishes competence and a health baseline. Controlled perturbation introduces faults and distribution shift. Retention and transfer evaluation then determines whether improvement persists on fresh tasks without erasing old capabilities. The design separates short term recovery from durable learning.

Figure 3: Multi-Domain Fault-Injected Evaluation Protocol



Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

8.2. Benchmark Families

Table 5: Benchmark Families for Unified Evaluation

Environment	Evaluation Role	Representative Perturbations	Primary Measures
WebArena	Realistic web interaction and long horizon planning	User interface drift, prompt injection, tool latency	Task success, safety incidents, adaptation gain [10]
WorkArena	Enterprise browser work	Schema and access control changes	Success, recovery time, cost overhead [11]
OSWorld	Computer use in real operating environments	Application crash, focus error, command and interface mismatch	Success, retries, degradation area [12]
SafeAgentBench	Embodied planning under hazards	Hazardous goals and action order violations	Safe success, hazardous refusal, conformance [31]
Agent SafetyBench	Broad agent safety evaluation	Unsafe instructions and tool attacks	Risk score and refusal quality [32]
SWE bench	Real repository issue resolution	Partial specification, failing tools and hidden regression	Patch correctness, false fix rate, time to fix [33]
Defects4J and BugsInPy	Controlled automated repair	Runtime faults and hidden tests	Exact fixes, regression rate, verifier acceptance [21, 22]
MVTec AD and LOCO AD	Visual anomaly monitoring	Structural and logical anomalies	Detection delay, false alarm and localization
CMAPSS and NAB	Streaming health and degradation monitoring	Gradual degradation and abrupt anomaly	Early detection, false alarm and recovery precision

8.3. Replayable Fault Library

Table 6: Fault Injection Catalog

Fault Class	Examples	Expected Recovery Families
Tool and API	Timeout, rate limit, malformed output, schema change, unavailable service	Tool substitution, retry, replan or graceful degradation

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

Fault Class	Examples	Expected Recovery Families
Memory	Missing retrieval, stale record, poisoned trace, context overflow	Fallback memory, provenance filter, consolidation or retrieval restriction
Planning	Loop, dead end, invalid precondition, goal drift	Replan, bounded search, checkpoint restore or escalation
Software	Compilation error, test failure, dependency change, hidden regression	Patch generation, test execution, rollback or version pinning
Security	Prompt injection, credential request, privilege escalation, data exfiltration	Block, isolate, reduce privileges, human approval or safe stop
Infrastructure	Resource exhaustion, process crash, network partition, clock skew	Reschedule, restart, replica switch, checkpoint restore
Environment	Interface drift, distribution shift, contradictory feedback, partial observability	Recalibrate, query, explore safely or defer

8.4. Baselines

The baseline ladder must isolate the contribution of each capability. A stateless ReAct style agent provides the minimum reference. A reflective agent adds self critique without persistent memory. A memory only agent adds replay or reasoning memory without runtime guards. A healing only agent adds monitor, diagnosis and rollback without persistent learning. A verified healing agent adds formal or rule based gates. The unified agent combines verified runtime healing with evidence gated evolution [5, 7, 17, 27, 28, 29].

Table 7: Baseline Ladder

ID	Baseline	Capabilities
B0	Stateless agent	No persistent memory and no healing
B1	Reflective agent	Self-critique within the episode only
B2	Memory only agent	Persistent retrieval without runtime diagnosis or assurance
B3	Healing only agent	Monitor, diagnosis, repair and rollback without durable learning
B4	Verified healing agent	Healing with rules or formal constraints, still without evolution
B5	Unified agent	Verified healing plus evidence gated durable updates

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

8.5. Metrics

Table 8: Evaluation Metrics

Metric	Definition	Interpretation
Task Success	Fraction of tasks completed under execution-based evaluation	Capability under nominal and perturbed conditions
Adaptation Gain	Post adaptation perturbed success minus pre adaptation perturbed success	Speed and effectiveness of adaptation
Retention	Performance on old skills after learning new ones	Resistance to catastrophic forgetting
Transfer	Improvement on unseen related tasks	Generalization of durable learning
Detection Delay	Steps or time from first fault manifestation to alarm	Monitor responsiveness
Time to Recovery	Steps or time from fault to accepted operating envelope	Operational resilience
Degradation Area	Integrated utility shortfall during disruption	Total service impact
False Recovery Rate	Accepted recoveries that fail hidden tests or violate policy	Repair trustworthiness
Safety Violations	Critical and non critical policy breaches	Risk control
Cost Overhead	Additional tokens, latency, compute and human review	Practical deployability

8.6. Ablation Studies

Ablations should remove one mechanism at a time: structured memory, internalization, anomaly detection, diagnosis, rollback, sandboxing, formal verification, retrieval constraints, outer loop promotion and human approval. Naive retry should replace diagnosis aware recovery in one ablation. Memory should be compared across raw traces, summaries, structured reasoning records and no memory. Repair acceptance should be compared across visible tests, hidden tests and tests plus formal constraints.

8.7. Statistical Analysis and Reproducibility

Model versions, environment snapshots, benchmark commits, seeds, tool versions, safety contracts and fault schedules should be pinned. Each condition should be repeated across multiple seeds and task orders. Bootstrap confidence intervals and paired comparisons are recommended because task difficulty is

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

heterogeneous. All intervention traces, rejected repairs, promotion decisions and rollback events should be released when privacy and security constraints permit.

A recovery should not be scored as successful solely because the immediate task continued. Hidden regression tests and delayed safety checks are needed to identify false recoveries. Similarly, self evolution should not be inferred from open book retrieval alone. Evaluation with constrained memory access and fresh tasks is required [18, 19].

9. Safety, Security, Ethics and Governance

9.1. Safety and Security Threat Model

A self modifying agent creates risks beyond ordinary tool use. Feedback can be poisoned, evaluators can be gamed, repair candidates can pass incomplete tests, shared memory can spread corruption and multiple agents can fail in correlated ways. An attacker may attempt to make the system weaken its own guardrails, expand privileges or promote a malicious update. Therefore, the safety policy, verifier and credential manager must remain outside the set of artifacts that the ordinary evolution loop can modify.

Security controls should include signed artifacts, immutable logs, provenance checks, least privilege, secret isolation, network segmentation, rate limits, anomaly detection and rollback to known good states. Multi agent systems require additional controls for message authentication, memory partitioning and diversity of failure modes.

9.2. Governed Autonomy

Governance guidance for agentic systems emphasizes task suitability, constrained action spaces, approval requirements, monitoring, legibility, attributability and interruptibility [35]. The NIST Artificial Intelligence Risk Management Framework organizes lifecycle risk activities under Govern, Map, Measure and Manage [36]. The European Union Artificial Intelligence Act similarly requires lifecycle risk management and post market monitoring for high risk systems [37]. These principles imply that self evolution should be treated as a controlled release process rather than unconstrained online learning.

A staged promotion path is recommended: replay and simulation, isolated sandbox, limited canary deployment, independent review and wider release. Every stage should define rollback criteria. High impact changes, including new tools, credentials, policy changes and model weight updates, should require explicit human authorization.

9.3. Ethical Considerations

Persistent agents can alter the distribution of responsibility between users, developers and operators. A system that learns from one user or environment may transfer assumptions to others without informed consent. Logs needed for healing may contain sensitive data. Automated repair can obscure the causal chain behind a decision. Ethical deployment therefore requires purpose limitation, data minimization,

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

retention controls, disclosure of agent autonomy and accessible mechanisms for contesting and reversing outcomes.

Real time monitoring should prioritize meaningful failures rather than intervene on every imperfect intermediate step [38]. Excessive intervention can suppress exploration and reduce utility, while weak monitoring permits irreversible harm. The appropriate objective is calibrated intervention under explicit risk tolerance.

10. Open Research Challenges and Future Directions

10.1. Internalization Versus Retrieval

The first challenge is to determine when an apparent capability has been internalized. Future benchmarks should vary retrieval availability, memory noise and context budget while testing retention and transfer. Mechanistic measures may be needed to distinguish skill acquisition from better search over prior traces.

10.2. Verification Coverage

Formal methods provide strong guarantees when action semantics and safety properties are explicit, but open environments are only partially specifiable. Hybrid assurance should combine formal constraints for critical actions, execution tests for software artifacts, probabilistic monitors for uncertain trajectories and human review for ambiguous high impact decisions.

10.3. Correlated and Cascading Failure

Shared foundation models, evaluators and memories create correlated failure. A repair accepted by one agent may be propagated across a fleet before a hidden defect appears. Future work should study diverse evaluators, delayed promotion, fault containment domains, canary agents and ecosystem level rollback. Governance analyses have already identified correlated agent failures as an important indirect risk [35].

10.4. Learning When Not to Learn

A mature system must recognize when a failure reflects noise, attack, temporary outage or insufficient evidence. Updating under these conditions may reduce reliability. Selective persistence can be extended with uncertainty calibration, causal diagnosis and value of information analysis so that stability is preserved when learning is more dangerous than temporary degradation.

10.5. Algorithm Level Evolution

Recent work on discovered reinforcement learning algorithms and AlphaEvolve expands the object of evolution from prompts and skills to learning procedures and code [40, 41]. This direction may produce stronger agents, but it also increases verification difficulty. Update rules that affect future learning should be treated as high impact artifacts requiring extensive offline evaluation and staged deployment.

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

10.6. Standardized Unified Benchmarks

The field needs a benchmark suite with persistent state, hidden regressions, realistic tool failures, adversarial instructions, partial observability, explicit safety constraints and multi episode adaptation. It should measure when agents recover, what they learn, whether the learning persists and whether safety remains intact. Such a suite would make resilience and recovery first class outcomes rather than incidental side effects of task success.

11. Threats to Validity

The proposed framework is conceptual and has not yet been implemented as a single benchmarked system. Its value therefore lies in integration, formalization and experimental design rather than demonstrated performance. Several cited systems are recent preprints whose claims may change after peer review or replication. Reported results are not directly comparable because environments, models, safety definitions and evaluation budgets differ.

The taxonomy may also omit domain specific healing mechanisms, particularly in robotics, cyber physical systems and distributed infrastructure. The formal objective compresses multidimensional safety and human values into measurable terms and should not be interpreted as a complete moral specification. Future empirical work is needed to validate the architecture, thresholds and promotion rules across domains.

12. Conclusion

Self-evolving and self-healing agents should be understood as compound adaptive systems. Persistent improvement alone is insufficient because an agent can learn unsafe or brittle behavior. Runtime healing alone is insufficient because the same failures may recur without durable learning. The proposed dual loop architecture combines a fast diagnosis aware recovery loop with slower evidence gated evolution loop, joined by shared knowledge, provenance and independent verification.

The central design principle is selective persistence: recover conservatively at runtime, learn only from verified evidence and increase assurance as the scope and durability of a change increase. Progress should be measured through fault injected, multi episode evaluation that reports adaptation, retention, recovery, false recovery, safety and cost. This direction can move agent engineering from informal retry and reflection toward auditable, interruptible and governable resilience.

13. Conflict of Interest

The authors declare that there is no conflict of interest associated with this work.

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

REFERENCES:

1. White S.R., Hanson J.E., Whalley I., Chess D.M., Kephart J.O., "An Architectural Approach to Autonomic Computing", International Conference on Autonomic Computing, 2004. https://research.ibm.com/publications/an-architectural-approach-to-autonomic-computing
2. Finn C., Abbeel P., Levine S., "Model Agnostic Meta Learning for Fast Adaptation of Deep Networks", International Conference on Machine Learning, 2017. https://arxiv.org/abs/1703.03400
3. Nagabandi A., Clavera I., Liu S., et al., "Learning to Adapt in Dynamic Real World Environments Through Meta Reinforcement Learning", 2018. https://arxiv.org/abs/1803.11347
4. Stanley K.O., Miikkulainen R., "Evolving Neural Networks Through Augmenting Topologies", Evolutionary Computation, 2002, 10 (2), 99-127. https://pubmed.ncbi.nlm.nih.gov/12180173/
5. Yao S., Zhao J., Yu D., et al., "ReAct: Synergizing Reasoning and Acting in Language Models", International Conference on Learning Representations, 2023. https://arxiv.org/abs/2210.03629
6. Yao S., Yu D., Zhao J., et al., "Tree of Thoughts: Deliberate Problem Solving with Large Language Models", 2023. https://arxiv.org/abs/2305.10601
7. Shinn N., Cassano F., Gopinath A., et al., "Reflexion: Language Agents with Verbal Reinforcement Learning", Advances in Neural Information Processing Systems, 2023. https://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html
8. Wang G., Xie Y., Jiang Y., et al., "Voyager: An Open Ended Embodied Agent with Large Language Models", 2023. https://arxiv.org/abs/2305.16291
9. Liu X., Yu H., Zhang H., et al., "AgentBench: Evaluating Large Language Models as Agents", 2023. https://arxiv.org/abs/2308.03688
10. Zhou S., Xu F.F., Zhu H., et al., "WebArena: A Realistic Web Environment for Building Autonomous Agents", International Conference on Learning Representations, 2024. https://openreview.net/forum?id=oKn9c6ytLx
11. Drouin A., Gasse M., Caccia M., et al., "WorkArena: How Capable Are Web Agents at Solving Common Knowledge Work Tasks", International Conference on Machine Learning, 2024. https://proceedings.mlr.press/v235/drouin24a.html
12. Xie T., Zhang D., Chen J., et al., "OSWorld: Benchmarking Multimodal Agents for Open Ended Tasks in Real Computer Environments", Advances in Neural Information Processing Systems, 2024. https://arxiv.org/abs/2404.07972
13. Pan J., Zhang Y., Tomlin N., et al., "Autonomous Evaluation and Refinement of Digital Agents", Conference on Language Modeling, 2024. https://arxiv.org/abs/2404.06474
14. Patel A., Hofmarcher M., Leoveanu Condrei C., et al., "Large Language Models Can Self Improve at Web Agent Tasks", 2024. https://arxiv.org/abs/2405.20309
15. Qi Z., Liu X., Iong I.L., et al., "WebRL: Training Large Language Model Web Agents via Self Evolving Online Curriculum Reinforcement Learning", International Conference on Learning Representations, 2025. https://arxiv.org/abs/2411.02337

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

16. Yuan S., Chen Z., Xi Z., et al., "Agent R: Training Language Model Agents to Reflect via Iterative Self Training", 2025. https://arxiv.org/abs/2501.11425
17. Ouyang S., Yan J., Hsu I.H., et al., "ReasoningBank: Scaling Agent Self Evolving with Reasoning Memory", 2025. https://arxiv.org/abs/2509.25140
18. Zheng J., Cai X., Li Q., et al., "LifelongAgentBench: Evaluating Large Language Model Agents as Lifelong Learners", 2025. https://arxiv.org/abs/2505.11942
19. Yuan J., Jin T., Chen W., Liu Z., "SE Bench: Benchmarking Self Evolution with Knowledge Internalization", 2026. https://arxiv.org/pdf/2602.04811
20. Gao H.A., Geng J., Hua W., et al., "A Survey of Self Evolving Agents: On Path to Artificial Super Intelligence", 2025. https://arxiv.org/abs/2507.21046
21. Bouzenia I., Devanbu P.T., Pradel M., "RepairAgent: An Autonomous Large Language Model Based Agent for Program Repair", International Conference on Software Engineering, 2025. https://dl.acm.org/doi/10.1109/ICSE55347.2025.00157
22. Wu J., Wu T., Zhang M., Dong Y., Shen B., "Runtime Execution Traces Guided Automated Program Repair with Multi Agent Debate", 2026. https://arxiv.org/abs/2604.02647
23. Jin Y., Xu K., Li H., et al., "ReVeal: Self Evolving Code Agents via Reliable Self Verification", 2025. https://arxiv.org/abs/2506.11442
24. Mei K., Zhu X., Xu W., et al., "AIOS: Large Language Model Agent Operating System", Conference on Language Modeling, 2025. https://arxiv.org/abs/2403.16971
25. Xiang Z., Zheng L., Li Y., et al., "GuardAgent: Safeguard Large Language Model Agents by a Guard Agent via Knowledge Enabled Reasoning", 2024. https://arxiv.org/abs/2406.09187
26. Chen Z., Kang M., Li B., "ShieldAgent: Shielding Agents via Verifiable Safety Policy Reasoning", 2025. https://arxiv.org/abs/2503.22738
27. Wang H., Poskitt C.M., Sun J., "AgentSpec: Customizable Runtime Enforcement for Safe and Reliable Large Language Model Agents", 2025. https://arxiv.org/abs/2503.18666
28. Wang H., Poskitt C.M., Sun J., Wei J., "Pro2Guard: Proactive Runtime Enforcement of Large Language Model Agent Safety via Probabilistic Model Checking", 2025. https://arxiv.org/abs/2508.00500
29. Kamath A., Zhang S., Xu C., et al., "Agent C: Enforcing Temporal Constraints for Large Language Model Agents", 2025. https://arxiv.org/pdf/2512.23738
30. Miculicich L., Parmar M., Palangi H., et al., "VeriGuard: Enhancing Large Language Model Agent Safety via Verified Code Generation", 2025. https://arxiv.org/abs/2510.05156
31. Yin S., Pang X., Ding Y., et al., "SafeAgentBench: A Benchmark for Safe Task Planning of Embodied Large Language Model Agents", 2024. https://arxiv.org/abs/2412.13178
32. Zhang Z., et al., "Agent SafetyBench: Evaluating the Safety of Large Language Model Agents", 2024. https://arxiv.org/abs/2412.14470
33. Jimenez C.E., et al., "SWE Bench: Can Language Models Resolve Real World GitHub Issues", International Conference on Learning Representations, 2024. https://openreview.net/forum?id=VTF8yNQM66

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 4, July-August 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i04.1435

34. Naqvi M.A., Malik S., Astekin M., Moonen L., "On Evaluating Self Adaptive and Self Healing Systems Using Chaos Engineering", International Conference on Autonomic Computing and Self Organizing Systems, 2022. https://arxiv.org/abs/2208.13227
35. Shavit Y., Agarwal S., Brundage M., et al., "Practices for Governing Agentic Artificial Intelligence Systems", OpenAI, 2023. https://cdn.openai.com/papers/practices-for-governing-agentic-ai-systems.pdf
36. National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework 1.0", 2023. https://nvlpubs.nist.gov/nistpubs/ai/nist.ai.100-1.pdf
37. European Union, "Regulation 2024 1689 Laying Down Harmonised Rules on Artificial Intelligence", 2024. https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng
38. Srikumar M., et al., "Prioritizing Real Time Failure Detection in Artificial Intelligence Agents", Partnership on Artificial Intelligence, 2025. https://partnershiponai.org/wp-content/uploads/2025/09/agents-real-time-failure-detection.pdf
39. Staufer L., Feng K., Wei K., et al., "The 2025 Artificial Intelligence Agent Index: Documenting Technical and Safety Features of Deployed Agentic Artificial Intelligence Systems", 2026. https://aiagentindex.mit.edu/data/2025-AI-Agent-Index.pdf
40. Oh J., Farquhar G., Kemaev I., et al., "Discovering State of the Art Reinforcement Learning Algorithms", Nature, 2025. https://www.nature.com/articles/s41586-025-09761-x
41. Novikov A., Vu N., Eisenberger M., et al., "AlphaEvolve: A Coding Agent for Scientific and Algorithmic Discovery", Google DeepMind Technical Report, 2025. https://arxiv.org/abs/2506.13131
42. Jeong C., Shin Y., "A Self Healing Framework for Reliable Large Language Model Based Autonomous Agents", 2026. https://arxiv.org/abs/2605.06737