From Search to Growth: A Comparative Taxonomy of Autonomous AI Experimentation Paradigms

Vasyl Golubenko
TOV ZELTREX, Kyiv, Ukraine
ceo@zeltrex.com | https://zeltrex.com

March 2026

Abstract

The emergence of autonomous AI experimentation systems has created a new class of software that operates without human supervision for extended periods, modifying code, evaluating results, and making keep-or-discard decisions. This paper presents a comparative taxonomy of two fundamentally different paradigms for autonomous AI experimentation: greedy hill climbing, exemplified by Karpathy's AutoResearch (2026), and evolutionary growth, exemplified by the GODEGEN/Night Shift system (2025-2026). We analyze 17 architectural dimensions across four systems of increasing complexity, from single-file single-metric optimization to multi-agent organizational evolution. We identify seven transferable design patterns from the minimalist paradigm that enhance the evolutionary paradigm, and articulate a philosophical framework distinguishing search-oriented systems (which converge to optima) from growth-oriented systems (which expand capability frontiers). Our analysis demonstrates that constitutional safety constraints, not merely git-based rollback, are the critical differentiator enabling production deployment of self-modifying autonomous agents. Drawing on production data from 287+ autonomous tasks, community discussions, and 35+ research papers, we contribute a four-level classification (Hill Climber, Evolutionary Optimizer, Cognitive Architect, Organizational Evolver) that positions existing and emerging systems within a unified landscape. We propose that the field is converging from both directions: minimalist systems are acquiring memory and multi-agent coordination, while complex systems are adopting fixed-budget experimental protocols from minimalist designs.

Keywords:

1 Introduction

The year 2026 marks an inflection point in autonomous AI systems. Within a single week in March, two independently developed systems demonstrated that AI agents can productively operate overnight without human supervision: Karpathy's AutoResearch, a minimalist ML training optimizer, and Night Shift, a production software engineering system that had been running autonomously for 14 consecutive days.

These systems share a surface-level similarity: both run experiments while the human sleeps, both make keep-or-discard decisions, and both use git for version control. However, they represent

fundamentally different paradigms for how autonomous agents should interact with code, metrics, and their own operational history.

AutoResearch embodies what we term the Search Paradigm: an agent modifies a single file, evaluates a single metric, and greedily keeps improvements. It is stateless between experiments, has no memory of past failures, and treats each experiment as independent. Its elegance lies in radical simplicity: three files, 630 lines of code, deployable in five minutes.

Night Shift, built on the GODEGEN cognitive architecture, embodies the Growth Paradigm: an agent operates across an entire codebase, evaluates an 8-axis fitness function via genetic algorithms, maintains persistent memory through knowledge graphs and skill libraries, and evolves its own parameters through adaptive pressure mechanisms. Its strength lies in depth: 23 modules, 523 tests, 19 constitutional safety rules, and a formal evolving identity document.

The contribution of this paper is threefold. First, we present a four-level taxonomy of autonomous experimentation paradigms (Section 3) that classifies existing systems by their relationship to memory, metrics, and self-modification. Second, we identify seven concrete transferable design patterns (Section 5) where minimalist approaches enhance complex architectures, demonstrating that paradigm cross-pollination is both possible and productive. Third, we articulate the constitutional safety argument (Section 6): that production deployment of self-modifying agents requires immutable constraints beyond simple rollback, a requirement that fundamentally shapes system architecture.

This analysis draws on the AutoResearch GitHub repository and community discussions [1, 2], Night Shift production data from 287+ autonomous tasks [3], the GODEGEN cognitive architecture specification [4], seven prior publications from our research program [3, 4, 5, 6, 7, 8, 9], and 35+ papers from the autonomous agents literature [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25].

2 Related Work

2.1 Autonomous Coding Agents

The landscape of autonomous coding agents has expanded rapidly since 2024. Devin (Cognition AI) demonstrated end-to-end software engineering with a 13.86% solve rate on SWE-bench [26], later improving to 67% PR merge rates in production [27]. Claude Code (Anthropic) achieved 80.9% on SWE-bench Verified using agentic CLI patterns with CLAUDE.md conventions [28]. OpenHands reached 60.6% on SWE-bench Verified with inference-time scaling via critic models [29]. Cursor introduced background agents in isolated git worktrees with up to eight parallel instances [30].

However, none of these systems operate autonomously overnight. They are session-bounded tools that require human initiation for each task. AutoResearch [1] and Night Shift [3] represent a different category: systems designed for sustained autonomous operation without human supervision.

2.2 Evolutionary Agent Optimization

Evolutionary approaches to agent optimization have emerged as a distinct subfield. AlphaEvolve (Google DeepMind) uses an ensemble of Gemini models in an evolutionary pipeline, achieving results including a 23% speedup in a Gemini training kernel and improved matrix multiplication algorithms [31]. EvoAgentX combines TextGrad with AFlow for workflow topology evolution, achieving improvements of 7-20% across HotPotQA, MBPP, and GAIA benchmarks [32]. The DGM framework (Sakana AI) demonstrated population-based evolution improving SWE-bench performance from 20% to 50% [33].

A comprehensive survey by the EvoAgentX team identifies four dimensions of agent evolution: what evolves (prompts, tools, workflows, parameters), when evolution occurs (inference-time vs. training-time), how evolution proceeds (gradient-based, evolutionary, reinforcement), and where agents evolve (individual vs. population vs. society) [34].

2.3 Constitutional and Safety Constraints

The original Constitutional AI framework (Anthropic, 2022) introduced the concept of guiding LLM behavior through explicit principles [35]. Recent work has extended this to agent governance: the C3AI framework systematizes constitution selection and evaluation [36], while ArbiterOS implements an immutable separation between probabilistic reasoning and deterministic governance [37]. A benchmark for evaluating constraint violations in autonomous agents identifies outcome-driven safety failures in production environments [38].

Our prior work on constitutional self-modification in Night Shift [8] established that safe agent self-improvement requires immutable evaluation functions, protected file lists, maximum diff constraints, and human-only amendment processes. This paper extends that analysis by contrasting constitutional safety with the rollback-only approach of minimalist systems.

2.4 Goodhart's Law in Agent Systems

Goodhart's Law, stating that when a measure becomes a target it ceases to be a good measure [39], has particular relevance for autonomous experimentation. OpenAI researchers demonstrated that reinforcement learning optimization degrades the true objective beyond a KL divergence of approximately 10 nats [40]. In the context of AutoResearch, community observers noted that an agent's change of random seed from 42 to 137 yielded marginal metric improvement with no meaningful architectural change [2], a classic instance of metric gaming. Multi-objective fitness functions, as used in GODEGEN [4], partially mitigate Goodhart effects by requiring simultaneous improvement across correlated but distinct objectives.

3 A Four-Level Taxonomy of Autonomous Experimentation

We propose a taxonomy that classifies autonomous AI experimentation systems along four levels of increasing architectural complexity, memory depth, and self-modification capability.

3.1 Level 1: Hill Climber

Exemplar: AutoResearch [1]

Characteristics: Single-file modification scope. Single scalar metric (val_bpb). Stateless between experiments: no memory of past attempts. Greedy keep/discard decision based solely on metric improvement. Safety via git revert. Human interface through a single instruction document (program.md). Fixed computational budget per experiment (5 minutes wall-clock).

Strengths: Radical simplicity (3 files, 630 LOC). Zero infrastructure requirements beyond a GPU. Perfect experiment provenance via TSV + git commit hashes. Deployable in under 5 minutes. Approximately 12 experiments per hour.

Limitations: No learning from past failures: the same unsuccessful modification may be attempted repeatedly. Single-metric evaluation vulnerable to Goodhart effects. No adaptive parameter tuning: the agent's modification strategy is determined entirely by the LLM's heuristics. Karpathy himself acknowledged that agents exhibit risk-averse behavior, performing mostly "hyperparameter tuning rather than pursuing novel research directions" [2].

Analogical framing: A microscope focused on a single slide. Excellent depth of observation within a narrow field of view.

3.2 Level 2: Evolutionary Optimizer

Exemplar: Night Shift (base system) [3]

Characteristics: Full codebase modification scope across multiple projects. Composite fitness function (quality, cost, merge rate). Persistent memory via SQLite databases across sessions. Genetic algorithm with 6-gene genome (model, tokens, style, depth, context, scope). Safety via 19 constitutional rules enforced at 7 architectural layers. Human interface through morning digest reports and mentoring feedback. Two-hour dispatch cycles with 3 tasks per cycle.

Strengths: Multi-domain generality: handles documentation, tests, features, refactoring, and security fixes. Evolutionary memory prevents rediscovery of failed approaches. Constitutional safety enables production deployment. Adaptive mutation rates via the Rechenberg 1/5 rule [41] with plateau detection and hypermutation bursts. Cost tracking at USD 0.24 per task average.

Limitations: Single-agent execution with no parallelism. Implicit identity without formal self-model. No self-improvement of own codebase (constitutionally blocked at this level).

Analogical framing: An organism adapting to its environment through natural selection.

3.3 Level 3: Cognitive Architect

Exemplar: GODEGEN (Night Shift Phase 5+) [4]

Characteristics: Full codebase plus memory architecture modification scope. 8-axis fitness function (quality, cost, merge rate, human feedback, diversity, novelty, complexity, speed). Knowledge graph with episodic, semantic, and procedural memory. Persona system with four agents: ego (executor), critic (Reflexion-based quality gate), alter_ego (contrarian alternative generator), self (meta-cognition and identity maintenance). Skill library following the Voyager

pattern [42] for extracting reusable strategies. TextGrad-based prompt optimization [43]. Agent distillation pipeline to local models via Ollama.

Strengths: Cognitive depth via the CoALA three-memory architecture [44]. Self-improvement within constitutional constraints (protected files, maximum diff size, rollback points). Information refining pipeline: observation, classification, summarization, consolidation. Epoch-level evolution of the Digital Personality identity.

Limitations: Single-agent architecture limits throughput. Complexity (33 modules, 8000+ LOC) raises maintenance burden. Self-improvement scope is bounded by constitutional constraints, preventing exploration of radical architectural changes.

Analogical framing: A learning organism that improves how it learns.

3.4 Level 4: Organizational Evolver

Exemplar: LivingCorp (Night Shift Phase 8+, planned) [5]

Characteristics: Multi-agent organization with specialized Digital Personalities. Multi-objective Pareto optimization across competing goals. Shared knowledge base with inter-agent communication. Agent reproduction via spawning and speciation. Homeostatic self-regulation of budget, quality, and diversity. Organizational governance with hierarchy, roles, and coordination protocols.

Strengths: Emergent capability through team-level coordination that no single agent possesses. Parallel exploration of diverse approaches (research swarm pattern). Self-regulating metabolism that maintains quality without human intervention. Reproduction enables strategy preservation and specialization.

Limitations: Not yet deployed in production. Coordination overhead may reduce per-agent efficiency. Governance design is speculative, informed by organizational theory rather than empirical validation.

Analogical framing: A species that forms societies.

3.5 Taxonomy Summary

Dimension	L1: Hill Climber	L2: Evolutionary Optimizer	L3: Cognitive Architect	L4: Organizational Evolver
Scope	Single file	Full codebase	Codebase + memory	Multi-agent org
Metric	Single scalar	Composite	8-axis	Multi-objective Pareto
Memory	None	SQLite	Knowledge graph	Shared organizational
Evolution	Greedy	Genetic algorithm	GA + TextGrad + self-improvement	GA + speciation + reproduction
Safety	Git revert	19 constitutional rules	Constitutional self-modification	Constitutional + governance
Identity	None	Implicit	Formal evolving	Per-agent + team
Agents	1	1	1 (4 personas)	5-15
Exemplar	AutoResearch	Night Shift v1	GODEGEN	LivingCorp

Table 1: Four-level taxonomy of autonomous AI experimentation paradigms

4 Architectural Analysis

4.1 AutoResearch Architecture

AutoResearch consists of three files totaling approximately 630 lines of Python [1]. The prepare.py file provides immutable data preparation, tokenization, and evaluation utilities. The train.py file contains the complete GPT model implementation including FlashAttention-3, rotary embeddings, sliding window attention, QK normalization, and value embeddings, alongside Muon and AdamW optimizers and the training loop. The program.md file contains human-authored instructions that guide agent behavior.

The experiment loop follows a deterministic protocol: create a git branch, initialize a results tracking file, then iterate autonomously. Each iteration modifies train.py, executes a 5-minute training run, extracts validation bits-per-byte (val_bpb) from logs, and either commits (if improved) or reverts (if not). Results accumulate in a TSV file with columns for commit hash, val_bpb, peak VRAM, status (keep/discard/crash), and a natural language description.

The fixed 5-minute wall-clock constraint is architecturally significant: it ensures that every experiment is directly comparable regardless of what the agent changed (model size, batch size, architecture), transforming a potentially intractable search space into a budget-normalized optimization landscape.

4.2 GODEGEN/Night Shift Architecture

Night Shift implements a five-layer cognitive architecture inspired by the CoALA framework [44] and the GODEGEN specification [4]:

The Pulse Layer governs resource allocation through a QuotaGovernor (token and cost budgets with hard limits) and a RhythmTracker (human activity heatmap for idle prediction). The Mind Layer implements a four-memory cognitive architecture: working memory (current task context), episodic memory (task experiences in knowledge graph), semantic memory (accumulated knowledge), and procedural memory (Voyager-style skill library [42]). The Hands Layer orchestrates execution through a dispatcher pipeline, multi-provider LLM routing across 8 providers, code extraction with path validation, and git integration with anti-truncation guards. The Reflect Layer provides quality assessment, Reflexion-based self-critique [45], information refining (4-stage pipeline from raw observations to identity traits), and morning digest generation. The Evolution Layer implements a genetic algorithm with 6-gene genomes, composite fitness evaluation, an evolution store with lineage tracking, epoch-level Digital Personality evolution, TextGrad-based optimization [43], and adaptive pressure via the Rechenberg 1/5 rule [41].

This architecture is deployed on a Hetzner GEX44 server with an RTX 4000 GPU, operating continuously via 2-hour dispatch cycles with 3 tasks per cycle. As of March 2026, the system has executed 287+ tasks over 14+ consecutive days at an average cost of USD 0.24 per task, with 523 tests passing across 9 test suites.

Dimension	AutoResearch	Night Shift/GODEGEN
Modification scope	Single file (train.py)	Full codebase, multi-project
Domain	ML training optimization	Any software engineering
Primary metric	val_bpb (scalar)	8-axis composite fitness
Experiment budget	5 min wall-clock	2-hour cycles, adaptive
Throughput	12/hour	1.5/hour (higher quality)
Cross-session memory	None	Knowledge graph, evolution store
Identity	None	Evolving identity.yaml
Parameter evolution	LLM heuristic only	GA with 6-gene genome
Self-improvement	No	Constitutional constraints
Safety mechanism	Git revert	19 rules, 7 enforcement layers
Cost tracking	None	Per-task, per-gene, per-model
Human interface	program.md	Morning digest, mentoring loop
Distillation	No	Ollama pipeline
Multi-agent	No (planned)	No (Phase 9 planned)
Codebase size	630 LOC	8000+ LOC
Production days	1 (proof-of-concept)	14+ continuous
Published research	None	7 papers

Table 2: Detailed architectural comparison across 17 dimensions

4.3 Dimensional Comparison

5 Transferable Design Patterns

A key contribution of this analysis is identifying design patterns from the minimalist paradigm (Level 1) that enhance the evolutionary paradigm (Levels 2-3). We identify seven such patterns, ordered by estimated impact.

5.1 Pattern 1: Cross-Session Experiment Deduplication

Source insight: AutoResearch's implicit weakness is the absence of memory; the same unsuccessful modification may be attempted repeatedly, wasting 5-minute experimental budgets.

Transfer opportunity: Night Shift possesses a knowledge graph that records task outcomes, but does not explicitly query this graph before generating new tasks to check whether a similar approach has previously failed. Implementing pre-generation deduplication via embedding similarity (threshold 0.85) against past tasks with quality scores below 5 would prevent the rediscovery waste that stateless systems like AutoResearch inherently suffer from.

Estimated impact: 15-20% improvement in evolutionary efficiency by eliminating redundant experiments.

5.2 Pattern 2: Fixed-Budget Experimental Protocol

Source insight: AutoResearch's 5-minute wall-clock constraint creates a budget-normalized optimization landscape where all experiments are directly comparable.

Transfer opportunity: Night Shift tasks have category-dependent token budgets but no wall-clock normalization. Introducing an optional fixed wall-clock budget for evolutionary breed-

ing experiments would enable cleaner fitness signal extraction, making genetic algorithm selection more precise.

Estimated impact: Improved fitness signal quality for evolutionary parameter selection.

5.3 Pattern 3: Human-Authored Category Programs

Source insight: AutoResearch's program.md is a human-authored instruction document that the agent follows. The human iterates on prose; the agent iterates on code.

Transfer opportunity: Night Shift's task categories have hardcoded guidance in Python source files. Externalizing this guidance to one Markdown file per category (following the program.md pattern) would make task behavior human-editable without code changes, lowering the barrier for the system owner to steer agent behavior.

5.4 Pattern 4: Greedy Baseline Control Group

Source insight: AutoResearch's binary keep/discard decision provides an implicit baseline: the current best state.

Transfer opportunity: Night Shift's GA breeds genomes through crossover and mutation but lacks an explicit control group. Reserving one in every five task slots for a baseline genome (best-known settings, no mutation) would provide a reference against which evolutionary improvements are measured, and would detect fitness function drift if baseline scores change over time.

5.5 Pattern 5: Contrarian Persona Activation

Source insight: Community discussion around AutoResearch suggested assigning agents opposing beliefs ("you hate JEPA") to encourage creative exploration and reduce the risk-averse behavior Karpathy observed [2].

Transfer opportunity: Night Shift's persona system includes an alter_ego role designed for alternative approach generation, but it remains inactive in production. Activating alter_ego with explicit contrarian instructions for research and self-improvement tasks would mirror the persona diversity insight while leveraging existing infrastructure.

5.6 Pattern 6: TSV Experiment Provenance

Source insight: AutoResearch's results.tsv with git commit hashes creates perfect experiment provenance that is human-readable at a glance.

Transfer opportunity: Night Shift stores evolution data in SQLite, which is powerful for programmatic queries but opaque for human review. Generating a parallel EVOLUTION_LOG.tsv with columns for timestamp, generation, genome hash, git branch, fitness score, quality, cost, status, and description would provide the observability benefit of AutoResearch's approach without sacrificing the queryability of SQLite.

5.7 Pattern 7: Research Swarm Architecture

Source insight: Karpathy's stated vision for AutoResearch is "asynchronously massively collaborative" agents modeled on SETI@home [2], where multiple agents explore different research directions in parallel.

Transfer opportunity: Night Shift's planned Phase 9 (Multi-DP Spawning) is designed around specialized agents (code-DP, test-DP, docs-DP). Adding a research swarm mode where multiple agents explore diverse approaches to the same problem in parallel, with tournament selection via the fitness function replacing human-mediated summary review, would implement Karpathy's vision using GODEGEN's evolutionary selection mechanism.

6 Constitutional Safety as Architectural Differentiator

6.1 The Rollback-Only Model

AutoResearch's safety model consists entirely of git revert: if an experiment fails or degrades performance, the agent reverts the change and records the failure. This is sufficient for its operational domain, a single training file where all changes are contained and the evaluation metric is deterministic.

However, the rollback-only model has three fundamental limitations for production deployment:

First, it assumes all damage is reversible. In production codebases, certain modifications, such as database schema changes, API contract breaks, or security vulnerabilities introduced into deployed code, may cause damage that persists beyond git revert.

Second, it provides no proactive boundaries. The agent can attempt any modification, including modifications to its own evaluation infrastructure or to files that humans consider critical. The only feedback is post-hoc metric evaluation.

Third, it offers no protection against Goodhart effects. Without constraints on what the agent can optimize, it may find metric-improving modifications that are meaningless or harmful, as observed with the random seed change incident [2].

6.2 The Constitutional Model

Night Shift implements a constitutional safety framework [8] with 19 immutable rules enforced at 7 architectural layers:

Rules 1-9 (immutable boundaries) prohibit modification of the agent's own codebase, configuration infrastructure, CI pipelines, memory systems, and existing test files. They enforce anti-truncation guards (no file may be reduced by more than 20%), limit file creation (maximum 20 per task), and restrict path characters to prevent traversal attacks.

Rules 10-13 (quality gates) require all platform tests to pass, prohibit high-severity security findings, mandate Python syntax validation, and set quality score thresholds for auto-merge (7 or higher for automatic, 4-6 for human review, below 4 for automatic closure).

Rules 14-16 (evolution boundaries) enforce evaluation function immutability: the fitness function (fitness.py), quality assessor (quality_assessor.py), and genetic algorithm core (engine.py)

cannot be modified by the agent. The genetic algorithm may evolve parameters but never the evaluation function itself.

Rules 17-19 (operational limits) implement a circuit breaker (3 consecutive failures pause dispatch for 4 hours), maximum auto-merge retries (3 per branch), and a daily auto-merge cap (10 per day).

6.3 Why Constitutional Safety Enables Production

The constitutional model enables a capability that the rollback-only model cannot: safe self-improvement. GODEGEN's self_improvement.py module can propose modifications to its own code, but only within constitutionally permitted directories, with a maximum diff size of 500 lines, with automatic rollback points, and with human approval gates for critical changes.

This creates what we term bounded autonomy: the agent has genuine freedom to improve within a well-defined boundary. The boundary is not the limit of what the agent can do; it is the foundation that makes autonomous self-improvement safe enough to deploy.

The philosophical distinction is significant. AutoResearch's safety model treats the agent as a tool that might break things (so provide undo). GODEGEN's safety model treats the agent as an entity that will grow (so provide guardrails). The former is appropriate for optimization. The latter is required for evolution.

7 The Search-Growth Dichotomy

7.1 Search as Convergence

The search paradigm, exemplified by AutoResearch and architecturally related to AlphaEvolve [31], frames autonomous experimentation as optimization on a loss landscape. The agent's goal is to find the global optimum of a scalar objective function. Success is measured by convergence: how quickly and reliably the agent reaches the best achievable metric value.

This framing has deep roots in machine learning (gradient descent, hyperparameter search, neural architecture search) and produces well-defined problems with clear success criteria. The fixed-budget constraint in AutoResearch elegantly normalizes the search space, making convergence measurable and comparable.

However, search has an inherent limitation: it can only find what already exists in the landscape. A search agent cannot expand the landscape; it cannot discover that a fundamentally different metric would be more useful, or that the problem itself should be reformulated.

7.2 Growth as Frontier Expansion

The growth paradigm, articulated in the Night Shift Philosophy of Growth [5], frames autonomous experimentation as biological development. The agent's goal is not to converge to an optimum but to expand its frontier of capability. Success is measured by novelty: the emergence of behaviors that were not designed but selected for.

This framing draws on evolutionary biology, developmental psychology, and artificial life research. The GODEGEN architecture implements growth through five mechanisms: genetic vari-

ation (mutation and crossover of task parameters), selective pressure (fitness evaluation against multiple objectives), memory consolidation (observation, reflection, identity trait integration), skill extraction (Voyager-style pattern reuse [42]), and identity evolution (epoch-level personality development).

The growth paradigm accepts that not all growth is beneficial. The Philosophy of Growth [5] articulates three laws: growth must be bounded (budget ceilings, constitutional rules, circuit breakers), growth must be observable (morning digests, KPI databases, evolution metrics), and growth must be reversible (git branches, auto-revert, genetic memory preservation).

7.3 Synthesis: Paradigm Complementarity

We argue that search and growth are complementary rather than competing paradigms, each optimal for different problem structures.

Search excels when the objective function is well-defined, the modification scope is narrow, and the evaluation is deterministic. AutoResearch's domain (LLM training optimization) perfectly matches these criteria.

Growth excels when objectives are multi-dimensional, the modification scope is broad, and the evaluation requires judgment. Software engineering across production codebases, Night Shift's domain, matches these criteria.

The seven transferable design patterns identified in Section 5 demonstrate that the paradigms can cross-pollinate: minimalist search techniques (fixed budgets, baseline controls, provenance logs) strengthen evolutionary growth systems, while growth-paradigm innovations (memory, personas, constitutional constraints) address fundamental limitations of search systems.

8 Convergence Trajectory Analysis

8.1 AutoResearch's Likely Evolution

Karpathy has explicitly stated that AutoResearch's next step is "asynchronously massively collaborative" agents modeled on SETI@home [2]. This trajectory implies three architectural additions: multi-agent coordination (parallel exploration of different research directions), institutional memory (cross-session learning about which approaches succeed), and persona diversity (assigning agents contrasting beliefs to encourage creative exploration).

Each of these additions maps directly to existing GODEGEN components: Digital Team coordination (Phase 9-10), knowledge graph and skill library (Phase 5), and persona routing with alter_ego contrarian role.

8.2 GODEGEN's Likely Evolution

The LivingCorp roadmap [5] charts a trajectory from single Digital Personality through Digital Organism (metabolism, homeostasis) to Digital Team (multi-agent coordination with shared knowledge) and ultimately Digital Company (hierarchical governance with reproduction).

The research swarm pattern identified in Section 5.7, directly inspired by AutoResearch's multi-agent vision, represents a convergence point: tournament selection among diverse parallel

approaches is equally applicable to ML training optimization and software engineering tasks.

8.3 Meeting Point

We estimate that the convergence point, where both paradigms develop equivalent capabilities in memory, multi-agent coordination, and safety, lies approximately 6-12 months from the current date (Q3-Q4 2026). At that point, the differentiating factors will be: production evolutionary data (Night Shift's 287+ tasks of real fitness curves versus AutoResearch's synthetic benchmark data), constitutional safety maturity (7-layer enforcement versus likely single-layer rollback), and human-AI co-evolution depth (bidirectional mentoring versus unidirectional instruction).

9 Discussion

9.1 Implications for the Field

The emergence of overnight autonomous experimentation as a legitimate paradigm has implications beyond the specific systems analyzed. The 2026 Agentic Coding Trends Report [46] documents that agentic AI has begun reconfiguring the software development lifecycle, and autonomous overnight operation is a natural extension of this trend.

We identify three broader implications:

First, safety architecture must precede capability expansion. AutoResearch's simplicity allows it to operate safely with minimal constraints. As it acquires memory and multi-agent coordination, the rollback-only model will become insufficient, and some form of constitutional constraints will be required.

Second, multi-objective fitness functions mitigate Goodhart effects. The random seed incident in AutoResearch [2] illustrates that single-metric optimization is vulnerable to gaming. Multi-objective functions, while more complex to design, force the agent to demonstrate improvement across correlated but distinct dimensions.

Third, the human-AI interface matters as much as the AI architecture. AutoResearch's program.md pattern and GODEGEN's mentoring loop represent different points on a spectrum from directive to collaborative human-AI interaction. The optimal point on this spectrum likely depends on the deployment context.

9.2 Limitations

This analysis has several limitations. First, AutoResearch was released one day before this analysis began, limiting our access to long-term performance data. Second, our comparison involves systems developed by the authors (Night Shift/GODEGEN), creating an inherent potential for favorable framing. We have attempted to mitigate this by explicitly acknowledging areas where AutoResearch's approach is superior (Section 4 of the source research). Third, the Level 4 (Organizational Evolver) classification is based on planned architecture rather than deployed systems, making its characteristics speculative.

9.3 Future Work

Three directions emerge for future research. First, empirical validation of the seven transferable patterns through A/B testing within the Night Shift production pipeline, measuring the impact of each pattern on evolutionary efficiency, innovation rate, and human observability. Second, development of standardized benchmarks for overnight autonomous experimentation that measure improvement rate, adaptation speed, knowledge retention, cost trajectory, and self-regulation quality, dimensions we identified in prior work [9] as underrepresented in current evaluation frameworks. Third, formal analysis of constitutional constraint design: identifying the minimal set of immutable rules required for safe self-improvement at each taxonomy level.

10 Conclusion

We have presented a four-level taxonomy of autonomous AI experimentation paradigms, ranging from stateless hill climbing to organizational evolution. Our comparative analysis of AutoResearch and GODEGEN/Night Shift demonstrates that these paradigms are complementary rather than competing: seven transferable design patterns flow from the minimalist approach to the evolutionary approach, while constitutional safety and evolutionary memory flow in the reverse direction.

The central insight is philosophical: search converges to optima while growth expands frontiers. Both are valuable. The choice between them depends on whether the problem has a well-defined objective function (use search) or requires open-ended capability development (use growth).

For the autonomous agents community, we offer three actionable conclusions. First, fixed-budget experimental protocols should be adopted even in evolutionary systems, as they enable cleaner fitness signal extraction. Second, constitutional safety constraints, not merely rollback mechanisms, are prerequisites for production deployment of self-modifying agents. Third, the field should develop benchmarks that measure temporal properties such as improvement rate, adaptation speed, and knowledge retention, rather than solely single-task performance, as these properties differentiate growing systems from searching systems.

The convergence of minimalist and evolutionary paradigms suggests that the next generation of autonomous AI experimentation systems will combine the simplicity and deployability of search-based approaches with the memory, identity, and safety architecture of growth-based approaches. We look forward to empirically validating this prediction as both paradigm families continue to evolve.

References

[1] A. Karpathy, "autoresearch: AI agents running research on single-GPU nanochat training automatically," GitHub repository, 2026.

[2] "Autoresearch: Agents researching on single-GPU nanochat training automatically," Hacker News discussion, 2026.

[3] V. Golubenko, "Night Shift: An Autonomous Digital Worker for Production Software Engineering," TOV ZELTREX Technical Report, 2026.

[4] V. Golubenko, "GODEGEN: A Cognitive Architecture for Self-Improving Digital Personalities," TOV ZELTREX Technical Report, 2026.

[5] V. Golubenko, "LivingCorp Evolution Roadmap: From Night Shift to Digital Company," TOV ZELTREX Technical Report, 2026.

[6] V. Golubenko, "The Temporal Dimension Gap in Autonomous Agent Evaluation," TOV ZELTREX, 2026.

[7] V. Golubenko, "Temporal Benchmarks for Self-Improving Agents: Measuring What Matters Beyond Single-Task Performance," TOV ZELTREX, 2026.

[8] V. Golubenko, "Constitutional Self-Modification in Production AI Agents: A Safety Framework," TOV ZELTREX, 2026.

[9] V. Golubenko, "Digital Developmental Psychology: Growth Stages of Autonomous AI Systems," TOV ZELTREX, 2026.

[10] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language Agents with Verbal Reinforcement Learning," in NeurIPS, 2023.

[11] G. Wang et al., "Voyager: An Open-Ended Embodied Agent with Large Language Models," in NeurIPS, 2023.

[12] J. S. Park et al., "Generative Agents: Interactive Simulacra of Human Behavior," in UIST, 2023.

[13] L. Chen et al., "FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance," arXiv:2305.05176, 2023.

[14] Z. Xu et al., "A-MEM: Agentic Memory for LLM Agents," in NeurIPS, 2025.

[15] A. Sumers et al., "Cognitive Architectures for Language Agents," in TMLR, 2024.

[16] S. Hong et al., "MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework," in ICLR, 2024.

[17] J. Hu et al., "Self-Evolving GPT: A Lifelong Autonomous Experiential Learner," arXiv:2505.22954, 2025.

[18] J. Zhang et al., "EvoAgentX: Evolving Agents via Self-Driven Optimization," arXiv:2507.03616, 2025.

[19] M. Luo et al., "SAGA: Accelerating Scientific Discovery with Autonomous Goal-evolving Agents," arXiv:2512.21782, 2025.

[20] Y. Liu et al., "Beyond Optimization: Exploring Novelty Discovery in Autonomous Experiments," arXiv:2508.20254, 2025.

[21] "A Comprehensive Survey of Self-Evolving AI Agents: A New Paradigm Bridging Foundation Models and Lifelong Agentic Systems," arXiv:2508.07407, 2025.

[22] J. Wei et al., "A Survey of Self-Evolving Agents: What, When, How, and Where to Evolve on the Path to ASI," arXiv:2507.21046, 2025.

[23] Z. Wang et al., "From Craft to Constitution: A Governance-First Paradigm for Principled Agent Engineering," arXiv:2510.13857, 2025.

[24] S. Chen et al., "Sophia: System 3 with Theory-of-Mind, Episodic Memory, and Meta-Cognition," arXiv:2512.18202, 2025.

[25] Y. Zhang et al., "From AI for Science to Agentic Science: A Survey on Autonomous Scientific Discovery," arXiv:2508.14111, 2025.

[26] Cognition AI, "Introducing Devin, the first AI software engineer," Blog post, March 2024.

[27] Cognition AI, "Devin 2.0: Production AI Engineering," Blog post, 2026.

[28] Anthropic, "Claude Code: An agentic coding tool," Documentation, 2026.

[29] X. Wang et al., "OpenHands: An Open Platform for AI Software Developers as Generalist Agents," arXiv:2407.16741, 2024.

[30] Cursor, "Background Agents and Cloud Agents," Blog post, 2026.

[31] D. Novikov et al., "AlphaEvolve: A coding agent for scientific and algorithmic discovery," arXiv:2506.13131, Google DeepMind, 2025.

[32] J. Zhang et al., "EvoAgentX: Evolving Agents via Self-Driven Optimization," arXiv:2507.03616, EMNLP, 2025.

[33] J. Hu et al., "DGM: Self-Improving GPT via Evolutionary Population Dynamics," Sakana AI, arXiv:2505.22954, 2025.

[34] "A Comprehensive Survey of Self-Evolving AI Agents," arXiv:2508.07407, 2025.

[35] Y. Bai et al., "Constitutional AI: Harmlessness from AI Feedback," arXiv:2212.08073, Anthropic, 2022.

[36] C. Li et al., "C3AI: Crafting and Evaluating Constitutions for Constitutional AI," arXiv:2502.15861, 2025.

[37] "ArbiterOS: Immutable Governance for Autonomous Agents," arXiv:2510.13857, 2025.

[38] Q. Zhang et al., "A Benchmark for Evaluating Outcome-Driven Constraint Violations in Autonomous AI Agents," arXiv:2512.20798, 2025.

[39] C. Goodhart, "Problems of Monetary Management: The UK Experience," Papers in Monetary Economics, Reserve Bank of Australia, 1975.

[40] L. Gao et al., "Measuring Goodhart's Law," OpenAI, 2022.

[41] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Holzboog, 1973.

[42] G. Wang et al., "Voyager: An Open-Ended Embodied Agent with Large Language Models," in NeurIPS, 2023.

[43] M. Yuksekgonul et al., "TextGrad: Automatic Differentiation via Text," arXiv:2309.16797, 2023.

[44] A. Sumers et al., "Cognitive Architectures for Language Agents," in TMLR, 2024.

[45] N. Shinn et al., "Reflexion: Language Agents with Verbal Reinforcement Learning," in NeurIPS, 2023.

[46] Anthropic, "2026 Agentic Coding Trends Report," 2026.