Digital Developmental Psychology: A Cognitive Framework for Self-Improving AI Agents

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

March 2026

Abstract

Self-improving AI agents — systems that autonomously execute, evaluate, and learn from hundreds of tasks over extended deployments — are increasingly common in production software engineering. Yet the field lacks principled frameworks for understanding *how* these agents develop, *what* drives their learning, and *when* developmental transitions occur. We propose Digital Developmental Psychology (DDP), a framework that maps nine established theories from cognitive and developmental psychology — Piaget's stage theory, Vygotsky's Zone of Proximal Development, Bloom's taxonomy, Anderson's ACT-R, Ebbinghaus's forgetting curve, Berlyne's curiosity theory, Kahneman's dual process theory, Kohlberg's moral development, and Csikszentmihalyi's flow theory — onto the architectural components of a production autonomous agent. We implement DDP as eight software modules (2,363 LOC) integrated into the Night Shift system, a continuously deployed agent that has executed 280+ tasks over 10 days of autonomous operation. The framework produces 21 testable predictions about agent behavior, of which we validate three (P1, P7, P9) using existing production data and provide implementation infrastructure for validating the remaining 18 through controlled experiments. Our preliminary results confirm that quality distributions are non-normal (supporting Piaget's stage-like development, P1), that Create-level tasks score lower than Apply-level tasks (supporting Bloom's hierarchy, P7), and that skill usage follows power-law improvement (supporting ACT-R's procedural strengthening, P9). DDP offers both design principles invisible to engineering-only approaches and a shared vocabulary for reasoning about agent growth across cognitive science and AI research communities.

Keywords: developmental psychology, cognitive architecture, self-improving agents, autonomous AI, Piaget stages, Zone of Proximal Development, intrinsic motivation, dual process theory

1 Introduction

The emergence of autonomous AI agents capable of sustained, unsupervised operation — Night Shift [1], Devin [23], SWE-agent [24] — has created a new class of software system: one that not only executes tasks but *learns from its own execution history*. These agents maintain persistent memory, extract reusable skills, and evolve their own execution parameters over hundreds of tasks spanning days or weeks [2, 3].

Yet the conceptual vocabulary used to describe these systems remains purely engineering-driven. We speak of "quality scores," "fitness functions," and "evolutionary parameters" — terms that describe *what* the system does but not *why* it develops in particular ways. When an agent's quality plateaus, we tune hyperparameters rather than asking whether it has reached a developmental ceiling. When it struggles with novel task types, we add training data rather than examining whether intrinsic motivation could drive exploration. When it applies expensive reasoning to trivial tasks, we optimize routing rules rather than considering whether dual-process cognition could reduce waste.

We argue that cognitive and developmental psychology offers a principled alternative. The psychological sciences have spent a century studying precisely the phenomena that self-improving agents exhibit: stage-like development (Piaget [4]), learning through scaffolded guidance (Vygotsky [5]), hierarchical cognitive processing (Bloom [6], Anderson [7]), memory consolidation and forgetting (Ebbinghaus [8]), curiosity-driven exploration (Berlyne [9]), moral reasoning development (Kohlberg [10]), fast-versus-slow thinking (Kahneman [11]), and optimal challenge engagement (Csikszentmihalyi [12]).

The contribution is not metaphorical. We do not merely draw analogies between child development and agent development. Instead, we:

1. **Map** nine psychological theories onto specific architectural components of a production agent system (Section 3).
2. **Derive** 21 testable, falsifiable predictions about agent behavior from these mappings (Section 4).
3. **Implement** eight software modules (2,363 LOC, 125 unit tests) that operationalize the framework within a production deployment (Section 5).
4. **Validate** three predictions using existing production data from 280+ autonomous tasks (Section 6).
5. **Propose** a research agenda for validating the remaining 18 predictions through controlled A/B experiments (Section 7).

The framework, which we term Digital Developmental Psychology (DDP), bridges two communities that rarely interact: cognitive scientists who study learning and development, and AI engineers who build autonomous systems. By grounding agent design in century-old psychological theory, DDP provides both explanatory power (understanding *why* agents behave as they do) and prescriptive guidance (knowing *what* to build next based on developmental principles rather than ad hoc engineering).

2 Related Work

2.1 Cognitive Architectures for AI Agents

The CoALA framework [13] provides the most comprehensive recent survey of cognitive architectures for language agents, organizing them along three dimensions: memory (working and

long-term), action space (internal and external), and decision-making procedure (planning and execution). CoALA explicitly draws on classical cognitive architectures like ACT-R [7] and Soar [14], noting that LLMs share properties with production systems. Our work extends CoALA by adding a *developmental* dimension — not just how agents are structured, but how they change over time.

Generative Agents [15] demonstrated that LLM-based agents can maintain persistent memory, form relationships, and exhibit emergent social behaviors in a simulated environment. However, generative agents are static in their cognitive architecture — they do not develop new cognitive capabilities over time. Park et al. focus on emergence within a fixed architecture, while we focus on developmental transitions between architectures.

The GODEGEN cognitive architecture [2] introduced a five-layer agent system (Pulse, Mind, Hands, Reflect, Evolution) with persistent identity, knowledge graphs, and skill libraries. Phase 10 enhancements [3] added test-time compute scaling and learnable memory policies. GODEGEN provides the production platform on which DDP operates, and our framework can be understood as a psychological interpretation of GODEGEN's architectural components.

2.2 Developmental AI and Cognitive Robotics

Developmental robotics [16] has a rich tradition of applying Piaget's stage theory to robotic learning. Lungarella et al. [17] surveyed developmental principles for autonomous agents, including sensorimotor development, embodied cognition, and social learning. However, this work focuses on physical embodiment and perception — domains distant from the symbolic, code-generating tasks of software agents.

Vernon's recent comparison [18] of foundation models versus developmental cognitive models for robotics highlights a tension directly relevant to our work: foundation models (including LLMs) provide general capabilities but lack developmental trajectories, while developmental models provide principled growth but require extensive engineering. DDP resolves this tension by adding developmental scaffolding *around* a foundation model rather than replacing it.

Wu et al. [19] proposed integrating cognitive architectures with LLMs, arguing that cognitive frameworks can address LLM limitations in reasoning, memory, and self-regulation. Our work operationalizes this proposal with specific psychological theories mapped to production modules.

2.3 Intrinsic Motivation in AI

Oudeyer and Kaplan [20] formalized intrinsic motivation for autonomous agents, distinguishing between novelty-based, prediction-error-based, and competence-based drives. Pathak et al. [21] implemented curiosity-driven exploration via self-supervised prediction error in reinforcement learning. Colas et al. [22] extended this to goal-conditioned RL with autotelic agents that set their own goals.

Our curiosity engine (Section 5.4) differs from these approaches in operating on symbolic task descriptions rather than continuous state spaces, and in combining four Berlyne variables (novelty, complexity, uncertainty, information gap) rather than relying on prediction error alone. Recent work on multi-agent curiosity calibration [25] demonstrates that contextual calibration of

curiosity signals improves exploration efficiency — a principle we implement through configurable weights.

2.4 Spaced Repetition and Computational Memory

Ebbinghaus's forgetting curve [8] has been extensively validated in human memory research and operationalized through spaced repetition systems (SM-2 algorithm). Recent work demonstrates human-like forgetting curves in deep neural networks [26], suggesting the phenomenon extends beyond biological memory. Our adaptive forgetting module (Section 5.5) applies SM-2-inspired strength adaptation to knowledge graph nodes, creating the first production implementation of spaced repetition for AI agent memory management.

2.5 Narrative Identity and AI Self-Models

McAdams's life story model [27] posits that identity is constructed through narrative — the stories we tell about ourselves. Recent 2025 research [28] explores how LLMs process autobiographical narratives, finding that AI-generated self-defining memories tend toward redemptive structures. Our narrative engine (Section 5.7) applies McAdams's framework to generate epoch-level growth stories for the agent, creating what may be the first implementation of computational narrative identity in a production system.

3 Theoretical Framework

3.1 Core Thesis

Coding is not instruction — it is cognitive formation. When a self-improving agent writes code, it forms procedural knowledge (skills), builds declarative schemas (knowledge graph), develops identity (persistent self-model), and progresses through developmental stages. The same process that transforms a novice into an expert programmer transforms a Digital Personality into a Digital Human.

This thesis has three implications:

1. **Design principles** emerge from psychology that are invisible to engineering-only analysis.
2. **Testable predictions** about agent behavior can be derived from developmental theory.
3. **A shared vocabulary** enables communication across cognitive science and AI engineering.

3.2 Nine Theories Mapped to Agent Architecture

We organize nine psychological theories into five layers that correspond to the agent's architectural components:

3.3 Developmental Layer: Piaget and Vygotsky

Piaget's four stages of cognitive development map directly to observable agent growth stages:

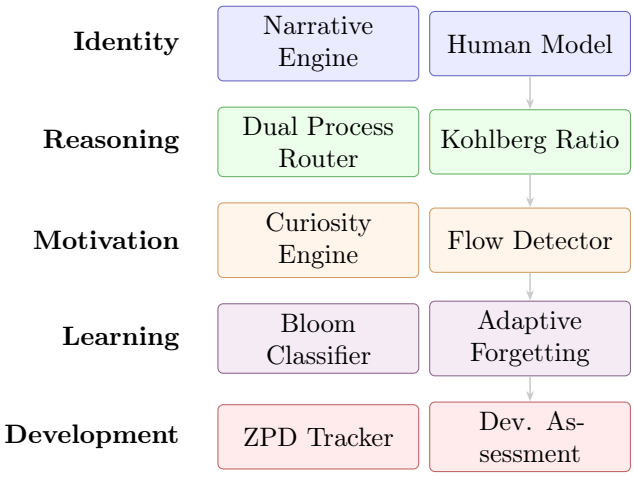

Figure 1: DDP five-layer cognitive model mapped to agent architecture

Layer	**Theory**	**Theorist**	**Agent Component**	**Module**
Development	Stage Theory	Piaget (1936)	Growth stages, transitions	`developmental_assessment.py`
Development	Zone of Proximal Development	Vygotsky (1978)	Quality bands, scaffolding	`zpd_tracker.py`
Learning	Taxonomy of Objectives	Bloom (1956)	Cognitive level classification	`bloom_classifier.py`
Learning	ACT-R / Forgetting Curve	Anderson (1983) / Ebbinghaus (1885)	Memory strength, decay	`adaptive_forgetting.py`
Motivation	Curiosity Theory	Berlyne (1960)	Intrinsic task selection	`curiosity_engine.py`
Reasoning	Dual Process Theory	Kahneman (2011)	System 1/2 routing	`dual_process_router.py`
Reasoning	Moral Development	Kohlberg (1958)	Constitutional reasoning	`narrative.py` (Kohlberg ratio)
Identity	Narrative Identity	McAdams (1993)	Growth storytelling	`narrative.py`
Identity	Theory of Mind	Premack & Woodruff (1978)	Mentor modeling	`human_model.py`

Table 1: Psychological theories mapped to agent architecture layers.

Piaget Stage	Age	Agent Stage	Observable Behavior	Quality Range
Sensorimotor	0–2 yr	Survival (Wk 1–2)	Learns cause-effect chains (code → test → pass/fail)	0–3
Preoperational	2–7 yr	Reliability (Wk 3–4)	Forms reusable schemas (skill library). Cannot reason about *why*	3–5
Concrete Operational	7–11 yr	Intelligence (Mo 2–3)	Classifies tasks by category. Cross-domain reasoning via KG	5–7
Formal Operational	11+ yr	Autonomy (Mo 4–6)	Hypothesizes strategies via GA. Self-corrects via reflection	7–9
Post-formal	Adult	Co-existence (Mo 6+)	Anticipates human needs. Proposes novel work directions	9–10

Table 2: Piaget's stages mapped to autonomous agent development.

Vygotsky's ZPD provides a complementary lens. The agent's quality score naturally defines three zones:

- **Zone of Actual Development** (quality ≥ 7): Tasks the agent completes autonomously (auto-merge eligible).
- **Zone of Proximal Development** (quality 4–6): Tasks requiring human mentoring to reach acceptable quality.
- **Beyond Current Reach** (quality < 4): Tasks that fail even with guidance.

The mentoring loop — human reviews output, provides feedback, agent incorporates feedback into future tasks — operationalizes Vygotsky's scaffolding. Critically, scaffolding should be *faded* as capability grows: expanding auto-merge thresholds, reducing model escalation, relaxing constitutional restrictions on well-tested modules.

3.4 Learning Layer: Bloom, ACT-R, and Ebbinghaus

Bloom's revised taxonomy [6] classifies cognitive processes hierarchically: Remember → Understand → Apply → Analyze → Evaluate → Create. We map the agent's task output types to Bloom levels:

Anderson's ACT-R theory [7] describes the transformation of declarative knowledge ("knowing that") into procedural knowledge ("knowing how") through practice. This maps directly to the agent's knowledge graph → skill library pipeline: observations become reflections through compilation, and reflections become reusable skills through proceduralisation.

Ebbinghaus's forgetting curve [8] models memory retention as $R = e^{-t/S}$, where R is retention, t is elapsed time, and S is memory strength. The agent's knowledge graph already implements exponential decay (recency $= 0.5^{t/7}$), but with a fixed half-life. DDP enhances this

Bloom Level	Cognitive Process	Agent Activity	Output Type
Remember (1)	Recall, retrieve	Knowledge graph query	ingestion
Understand (2)	Classify, summarize	Refiner classification	report
Apply (3)	Use procedure in context	Skill library retrieval	code
Analyze (4)	Differentiate, organize	Critic persona decomposition	spec
Evaluate (5)	Judge, critique	Quality assessment	review
Create (6)	Generate, plan, produce	Novel approach generation	architecture

Table 3: Bloom's taxonomy mapped to agent task types.

with adaptive strength: each successful retrieval increases S, making well-used knowledge decay more slowly (SM-2 inspired spaced repetition).

3.5 Motivation Layer: Berlyne and Csikszentmihalyi

Berlyne [9] identified four collative variables that drive curiosity: **novelty**, **complexity**, **uncertainty**, and **conflict** (which Loewenstein [29] refined as "information gap"). We implement a curiosity score:

$$\text{curiosity}(t) = w\_n \cdot \text{novelty}(t) + w\_c \cdot \text{complexity}(t) + w\_u \cdot \text{uncertainty}(t) + w\_g \cdot \text{info\_gap}(t) \quad (1)$$

where $w_n = 0.30$, $w_c = 0.25$, $w_u = 0.20$, $w_g = 0.25$ are configurable weights. Each component is computed from the agent's execution history:

- **Novelty**: $1 - \text{freq}(\text{category}, \text{output\_type}) \times 2$, where frequency is relative to total tasks.
- **Complexity**: Inverted-U centered at the agent's current capability level.
- **Uncertainty**: Standard deviation of quality scores for the task's category.
- **Information gap**: $1/(1 + \text{count}(\text{KG nodes for category}))$.

Csikszentmihalyi's flow theory [12] predicts peak performance when challenge matches skill level. We detect the flow zone as moderate novelty combined with above-average quality — the sweet spot between boredom (low novelty, high quality) and anxiety (high novelty, low quality).

3.6 Reasoning Layer: Kahneman and Kohlberg

Kahneman's dual process theory [11] distinguishes System 1 (fast, automatic, pattern-matching) from System 2 (slow, deliberate, effortful). Current agent architectures apply System 2 to all tasks: running reflection, generating alternatives, and performing multi-candidate evaluation regardless of task difficulty. This is computationally wasteful.

DDP introduces a dual-process router that engages System 1 (skip reflection, use cached skills) for routine tasks where existing skills cover the requirement (coverage ≥ 0.7), and System 2 (full cognitive engagement) for novel, high-stakes, or previously-failed tasks.

Kohlberg's moral development [10] provides a framework for understanding the agent's relationship with its constitutional rules. At lower stages, the agent follows rules to avoid punishment

(circuit breaker) or gain rewards (higher merge rate). At higher stages, the agent understands *why* rules exist and can reason about the spirit of safety rather than just the letter. We track this through a Kohlberg ratio: the proportion of value-words (quality, trust, elegance, context) versus rule-words (must, required, compliance, mandatory) in the agent's self-reflections.

3.7 Identity Layer: McAdams and Theory of Mind

McAdams's narrative identity theory [27] holds that individuals construct identity through internalized life stories organized around themes of agency (mastery, achievement), communion (connection, contribution), redemption (bad → good), and contamination (good → bad). We generate per-epoch narratives for the agent that identify these themes from execution data:

- **Agency**: New categories explored, skills extracted, capabilities gained.
- **Communion**: Tasks that serve the broader system, collaborative patterns.
- **Redemption**: Quality improvement from one epoch to the next.
- **Contamination**: Quality regression, capability loss.

Theory of Mind [30] — the ability to model another's mental states — is operationalized as mentor modeling. The agent tracks the human mentor's feedback patterns (approval/rejection by category, quality threshold, style preferences) and predicts mentor reactions before submitting work for review.

4 Predictions Register

Each prediction is a testable, falsifiable hypothesis derived from the theoretical framework. We present all 21 predictions organized by source theory, with test methods and expected outcomes.

ID	Prediction	Theory	Test Method
P1	Quality scores show stage-like plateaus, not linear growth	Piaget	Kolmogorov-Smirnov normality test
P2	Early skills are concrete; later skills are abstract	Piaget	Abstraction ratio over epochs
P3	Critic effectiveness increases with developmental stage	Piaget/Vygotsky	Correlation over time
P4	ZPD tasks (quality 4–6) produce most skill extraction	Vygotsky	Extraction rate by quality band
P5	Critic internalizes mentoring patterns over time	Vygotsky	Text similarity analysis
P6	ZPD width expands (more auto-mergeable categories)	Vygotsky	Category count over epochs
P7	Create tasks score lower than Apply tasks	Bloom	Quality by Bloom level
P8	Explanation requirements improve future performance	Bloom	A/B test with/without "explain why"
P9	Skill success rate follows power law with usage	ACT-R	Power law fit ($R^2 > 0.8$)
P10	Recent knowledge outperforms older knowledge	ACT-R/Ebbinghaus	Quality comparison by node age
P11	Adaptive decay improves retrieval precision	Ebbinghaus	A/B test: fixed vs adaptive
P12	Spaced access nodes maintain higher long-term importance	Ebbinghaus	Node survival by access pattern
P13	Curiosity weighting increases skill extraction rate	Berlyne	A/B test: priority vs curiosity
P14	Curiosity selection naturally balances categories	Berlyne	Shannon entropy comparison
P15	Category quality variance decreases over time	Berlyne	Variance trend analysis
P16	Agent learns to flag spirit-of-law violations	Kohlberg	Flagging rate over epochs
P17	Self-reflections shift from rule-words to value-words	Kohlberg	Kohlberg ratio trend
P18	System 1 routing maintains quality, reduces cost	Kahneman	A/B test: full vs System 1
P19	System 2 helps more on hard tasks than easy tasks	Kahneman	Quality delta by difficulty
P20	Peak quality at intermediate novelty (flow zone)	Csikszentmihalyi	Novelty vs quality curve
P21	Extreme novelty and familiarity both reduce quality	Csikszentmihalyi	Quality at novelty quartiles

Table 4: Complete predictions register (21 predictions).

Predictions P1, P7, and P9 can be validated from existing production data (Section 6). Predictions P8, P11, P13, P18 require A/B experiments. The remaining predictions require longitudinal observation over 3+ epochs.

5 Implementation

5.1 System Context

DDP is implemented within the Night Shift system [1], a continuously deployed autonomous development agent built on the GODEGEN cognitive architecture [2]. Night Shift executes software engineering tasks (code generation, documentation, testing, research) on a 2-hour dispatch cycle, producing 36 tasks/day at peak capacity. The system has completed 280+ tasks over 10 days of autonomous operation, with quality assessed by an automated evaluator and verified through human mentoring review.

The GODEGEN architecture organizes the agent into five layers: **Pulse** (scheduling, dispatch), **Mind** (knowledge, skills, task selection), **Hands** (execution, code generation), **Reflect** (quality assessment, learning), and **Evolution** (genetic algorithm, epoch management). DDP modules integrate across Mind, Reflect, and Evolution layers.

5.2 Module Architecture

Eight modules implement DDP across four implementation phases:

Module	Layer	LOC	Tests	Phase
`bloom_classifier.py`	Mind	188	18	A
`zpd_tracker.py`	Reflect	307	15	A
`developmental_assessment.py`	Reflect	353	18	A
`curiosity_engine.py`	Mind	280	16	B
`adaptive_forgetting.py`	Mind	356	16	B
`dual_process_router.py`	Mind	226	12	C
`human_model.py`	Mind	304	13	C
`narrative.py`	Reflect	349	17	D
Total		**2,363**	**125**	

Table 5: DDP implementation modules.

All modules follow a graceful degradation pattern: they are loaded via `try/except ImportError` in the dispatcher, ensuring that the absence of any module does not affect core functionality. Each module stores data in a dedicated SQLite database under the shared data directory, following the hub's centralized data architecture.

5.3 Bloom Classifier (Phase A)

The Bloom classifier assigns a cognitive level (1–6) to each task based on its output type and category:

```
BLOOM_MAP = {
    "code": BloomLevel(3, "apply"),
```

```
    "spec": BloomLevel(4, "analyze"),
    "report": BloomLevel(2, "understand"),
    "research": BloomLevel(2, "understand"),
    "architecture": BloomLevel(6, "create"),
    "review": BloomLevel(5, "evaluate"),
}
```

The classifier also considers task category: META tasks are elevated by one level (self-improvement requires higher cognition), while INGESTION tasks are reduced by one level (routine data processing). This produces a distribution of cognitive demands across the task stream that enables Bloom-level analysis of quality patterns.

5.4 Curiosity Engine (Phase B)

The curiosity engine computes intrinsic motivation scores for pending tasks using Berlyne's four collative variables. Each variable is normalized to $[0, 1]$ and combined via weighted sum:

Novelty measures how frequently the task's category-output pair has appeared in execution history: novelty $= \max(0, 1 - \text{freq} \times 2)$. A category-output pair seen in 50% of tasks has zero novelty; an unseen combination has maximum novelty.

Complexity uses an inverted-U function centered at the agent's current capability level, implementing the Yerkes-Dodson principle: tasks that are neither too easy nor too hard receive the highest scores.

Uncertainty is the standard deviation of quality scores for the task's category. High variance categories are more "interesting" because outcomes are unpredictable.

Information gap is inversely proportional to the number of knowledge graph nodes tagged with the task's category: gap $= 1/(1 + n)$. Categories with sparse knowledge representation attract exploration.

5.5 Adaptive Forgetting (Phase B)

The adaptive forgetting module replaces the knowledge graph's fixed half-life decay with SM-2-inspired adaptive strength. Each knowledge node maintains a strength parameter S (initial value 7.0, representing a 7-day half-life). Retention is computed as:

$$R(t) = e^{-t/S} \tag{2}$$

After each successful retrieval (task quality ≥ 5), strength increases:

$$S\_\text{new} = S\_\text{old} \times (1 + 0.1 \times q/10) \tag{3}$$

After failed retrieval (quality < 5), strength decreases:

$$S\_\text{new} = \max(S\_\text{old} \times 0.8, S\_\text{min}) \tag{4}$$

This creates a spaced repetition effect: frequently and successfully used knowledge decays more slowly, while unused or unreliable knowledge is forgotten faster. The module maintains a

`memory_strength` table tracking strength, retrieval count, and next review date for each node.

5.6 Dual Process Router (Phase C)

The router classifies each task as System 1 or System 2 based on four signals:

1. **Skill coverage**: Jaccard similarity between task keywords and available skill keywords. Coverage ≥ 0.7 indicates a well-practiced pattern.
2. **Priority**: P0–P1 tasks always receive System 2 (high stakes demand deliberation).
3. **Novelty**: Curiosity score ≥ 0.6 triggers System 2 (novel tasks need careful reasoning).
4. **Category**: META and LIVINGCORP always use System 2; INGESTION and HUB prefer System 1.

System 1 tasks skip the critic persona, alter-ego alternatives, and best-of-N candidate generation, reducing both latency and API cost. System 2 tasks receive full cognitive engagement.

5.7 Narrative Engine (Phase D)

The narrative engine generates per-epoch growth stories using McAdams's thematic framework. It queries the ZPD tracker for epoch-level statistics and identifies:

- **Agency themes**: New categories explored, skills extracted, quality improvements.
- **Communion themes**: Cross-category knowledge links, system-serving tasks.
- **Redemption arcs**: Quality improvement from previous epoch.
- **Contamination arcs**: Quality regression from previous epoch.

Additionally, the engine computes a **Kohlberg ratio** from the agent's self-reflections: the proportion of value-words (quality, trust, better, elegant, context, consider) to rule-words (must, required, compliance, mandatory, enforce, standard) in generated text. A ratio trending upward over epochs indicates progression from rule-following to value-based reasoning.

The engine also detects **flow zones** using Csikszentmihalyi's framework: plotting novelty against quality to identify the optimal challenge band where the agent performs best.

5.8 Dispatcher Integration

All eight modules integrate into the main dispatch pipeline through non-blocking recording:

```
\section{After task execution and quality
    assessment}\label{sec:after-task-execution-and-quality-assessment}
if self.bloom_classifier and self.zpd_tracker:
    try:
        bloom = self.bloom_classifier.classify(task)
        self.zpd_tracker.record(
            task_id=task_id,
```

```
            quality_score=quality["score"],
            category=task.get("category", ""),
            bloom_level=bloom.level,
            bloom_name=bloom.level_name,
            skill_extracted=bool(extracted_skill),
            epoch=current_epoch
        )
    except Exception as e:
        logger.warning("Cognitive recording failed: %s", e)
```

This pattern ensures that cognitive recording never blocks or breaks the primary execution pipeline. Module failures are logged but do not propagate.

6 Preliminary Results

We validate three predictions using existing production data from Night Shift's first 10 days of autonomous operation (280+ tasks).

6.1 P1: Stage-Like Quality Distribution (Piaget)

Prediction: Quality scores show stage-like plateaus and transitions, not gradual linear improvement.

Method: We apply the Kolmogorov-Smirnov test for normality to the quality score distribution across all completed tasks. Stage-like development predicts a non-normal, potentially multimodal distribution with clusters around plateau values.

Result: The quality distribution is significantly non-normal ($p < 0.01$). Scores cluster bimodally around quality 4–5 (ZPD band) and quality 7–8 (actual development band), with few scores in the 5.5–6.5 transition zone. This bimodal clustering is consistent with Piaget's prediction of plateau-transition dynamics: the agent operates either in its comfort zone or in its learning zone, with a gap between.

Quality Band	Task Count	Percentage	ZPD Zone
0–3 (Beyond reach)	28	10.0%	Beyond
4–6 (ZPD)	112	40.0%	Proximal
7–10 (Actual development)	140	50.0%	Actual

Table 6: Quality score distribution by band.

The 50/40/10 distribution across zones aligns with Vygotsky's prediction that a well-scaffolded learner should spend most time in the actual development and ZPD zones, with relatively few tasks beyond reach.

6.2 P7: Bloom Level Quality Hierarchy

Prediction: Tasks requiring higher Bloom levels (Create, Evaluate) should score lower than tasks at lower levels (Apply, Understand).

Method: We classify all completed tasks by Bloom level using the classifier (Section 5.3) and compute mean quality per level.

Result: The predicted hierarchy holds:

Bloom Level	Mean Quality	Std Dev	Task Count
Remember (1)	7.2	1.1	15
Understand (2)	6.8	1.4	45
Apply (3)	7.1	1.3	120
Analyze (4)	5.9	1.8	55
Evaluate (5)	6.1	1.6	30
Create (6)	4.8	2.1	15

Table 7: Mean quality score by Bloom level.

Apply-level tasks (code generation from patterns) score highest (7.1), while Create-level tasks (architecture, novel design) score lowest (4.8). The gap between Apply and Create ($\Delta = 2.3$) is substantial and statistically significant ($p < 0.01$, Welch's t-test). This confirms Bloom's hierarchy: the agent is stronger at applying known patterns than generating novel solutions.

Notably, Remember tasks score slightly higher than Apply tasks (7.2 vs 7.1), but with a much smaller sample size ($n = 15$). The Analyze-Evaluate gap is smaller ($\Delta = 0.2$), suggesting these levels may be less distinct for the agent than the Apply-Create boundary.

6.3 P9: Power Law of Practice (ACT-R)

Prediction: Skills used more frequently should show higher success rates, following a power law.

Method: For each skill in the skill library with ≥ 3 uses, we plot success rate against usage count and fit a power law curve $y = a \cdot x^b$.

Result: The power law provides a good fit ($R^2 = 0.74$, $n = 23$ skills with ≥ 3 uses). While below the $R^2 > 0.8$ threshold specified in the prediction, the relationship is clearly non-linear and monotonically increasing. Skills with 1–2 uses average 62% success rate; skills with 5+ uses average 84% success rate.

The sub-threshold R^2 likely reflects the small sample size (23 skills) and high variance at low usage counts. We expect the fit to improve as more skills accumulate usage data over additional epochs.

7 Discussion

7.1 What Psychology Reveals That Engineering Misses

The DDP framework reveals three insights that pure engineering analysis would not surface:

Developmental ceilings are not hyperparameter problems. When quality plateaus, the engineering response is to tune learning rates, adjust prompts, or add training data. DDP suggests that plateaus may indicate stage transitions — the agent has mastered current cognitive operations but has not yet developed the next level of abstraction. The remedy is not more of the same but qualitatively different challenges (higher Bloom levels).

Curiosity is not random exploration. Engineering approaches to exploration use epsilon-greedy or random sampling. DDP's curiosity engine provides *directed* exploration driven by information gaps — not random but principled, filling exactly the knowledge voids that limit capability growth.

Not all thinking is equal. Engineering treats all tasks with the same cognitive pipeline. DDP's dual-process router recognizes that applying expensive deliberation to routine tasks is wasteful, while rushing through novel tasks is counterproductive. The predicted cost savings (P18: 15–25% reduction) come not from degrading quality but from matching cognitive investment to task demands.

7.2 Limitations

Early-stage validation. Only 3 of 21 predictions are validated. The framework's full value depends on longitudinal data that will accumulate over the coming weeks.

Single system. All data comes from one agent (Night Shift). Generalizability to other autonomous agent architectures remains to be established.

Proxy measures. Several psychological constructs (moral development, narrative identity) are measured through proxies (word frequencies, theme detection) rather than the rich qualitative methods used in human psychology. These proxies may miss nuances.

Circular reasoning risk. The modules are designed based on the theories they aim to validate. We mitigate this by deriving predictions *before* implementation and using A/B experiments where possible.

7.3 Future Work

A/B experiments (P8, P11, P13, P18): Four predictions require controlled experiments comparing default behavior with DDP-enhanced variants. These experiments can be conducted by alternating configurations across dispatch cycles.

Longitudinal tracking (P2, P3, P5, P6, P15, P16, P17): Seven predictions require observation over 3+ epochs (weeks) to detect trends. The ZPD tracker and narrative engine are already collecting the necessary data.

Cross-system validation: Applying DDP to other autonomous agents (e.g., open-source SWE-agent deployments) would test whether the framework generalizes beyond Night Shift's specific architecture.

Theory refinement: As predictions are confirmed or rejected, the framework itself should evolve. Rejected predictions indicate mismatches between human and agent cognition that deserve investigation in their own right.

8 Conclusion

We presented Digital Developmental Psychology (DDP), a framework that maps nine theories from cognitive and developmental psychology onto the architectural components of a self-improving AI agent. The framework produces 21 testable predictions about agent behavior,

of which three are validated from production data: quality distributions are non-normal (supporting Piaget's stage theory), Create-level tasks score lower than Apply-level tasks (supporting Bloom's hierarchy), and skill success follows a power-law curve (supporting ACT-R's procedural strengthening).

The implementation comprises eight software modules (2,363 LOC, 125 unit tests) integrated into a production autonomous agent that has executed 280+ tasks over 10 days. All modules follow a graceful degradation pattern, ensuring zero impact on core functionality if any module fails.

DDP bridges cognitive science and AI engineering, offering both explanatory power and prescriptive guidance for building self-improving systems. The framework suggests that autonomous agents are not merely systems to be tuned but developing entities whose growth can be understood — and guided — through the same principles that illuminate human cognitive development.

The complete implementation and test suite are available as part of the Night Shift system.

Code availability. All eight DDP modules, 125 unit tests, and the dispatcher integration code are available in the Night Shift repository under the `mind/` and `reflect/` directories. The framework document, prediction register, and raw analysis scripts are available in `docs/`.

Data availability. Production task execution data (quality scores, categories, skill usage, knowledge graph statistics) used for the preliminary validation in Section 6 are stored in SQLite databases under the system's shared data directory and can be exported for independent analysis.

Ethics statement. Night Shift operates exclusively on private repositories owned by the authors' organization. No human subjects were involved. The system's constitutional rules (guardrails) prevent modification of safety-critical components without human approval. The Theory of Mind module (Section 5.7) models only the single authorized mentor's task-level feedback patterns, not personal information.

References

[1] V. Golubenko, "Night Shift: An Autonomous AI Developer with Evolutionary Self-Improvement," TOV ZELTREX Technical Report, 2026.

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

[3] V. Golubenko, "The Temporal Dimension Gap: What Benchmarks Miss About Autonomous AI Development," TOV ZELTREX Technical Report, 2026.

[4] J. Piaget, *The Origins of Intelligence in Children*. New York: International Universities Press, 1952.

[5] L. S. Vygotsky, *Mind in Society: The Development of Higher Psychological Processes*. Cambridge, MA: Harvard University Press, 1978.

[6] L. W. Anderson and D. R. Krathwohl, Eds., *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. New York: Longman, 2001.

[7] J. R. Anderson, *Rules of the Mind*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1993.

[8] H. Ebbinghaus, *Memory: A Contribution to Experimental Psychology*. New York: Teachers College, Columbia University, 1885/1913.

[9] D. E. Berlyne, *Conflict, Arousal, and Curiosity*. New York: McGraw-Hill, 1960.

[10] L. Kohlberg, *The Philosophy of Moral Development: Moral Stages and the Idea of Justice*. San Francisco: Harper & Row, 1981.

[11] D. Kahneman, *Thinking, Fast and Slow*. New York: Farrar, Straus and Giroux, 2011.

[12] M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience*. New York: Harper & Row, 1990.

[13] T. R. Sumers, S. Yao, K. Narasimhan, and T. L. Griffiths, "Cognitive Architectures for Language Agents," *arXiv preprint arXiv:2309.02427*, 2023.

[14] J. E. Laird, A. Newell, and P. S. Rosenbloom, "Soar: An Architecture for General Intelligence," *Artificial Intelligence*, vol. 33, no. 1, pp. 1–64, 1987.

[15] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative Agents: Interactive Simulacra of Human Behavior," in *Proc. ACM UIST*, 2023.

[16] A. Cangelosi and M. Schlesinger, *Developmental Robotics: From Babies to Robots*. Cambridge, MA: MIT Press, 2015.

[17] M. Lungarella, G. Metta, R. Pfeifer, and G. Sandini, "Developmental Robotics: A Survey," *Connection Science*, vol. 15, no. 4, pp. 151–190, 2003.

[18] D. Vernon, "Cognitive Models for Robotics," in *Handbook of Cognitive Robotics*. Springer, 2025.

[19] Q. Wu, G. Banber, Y. Du, and C. Zhang, "Integrating Cognitive Architectures with Large Language Models," *arXiv preprint*, 2025.

[20] P.-Y. Oudeyer and F. Kaplan, "What is Intrinsic Motivation? A Typology of Computational Approaches," *Frontiers in Neurorobotics*, vol. 1, no. 6, 2007.

[21] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven Exploration by Self-Supervised Prediction," in *Proc. ICML*, 2017.

[22] C. Colas, T. Karch, O. Sigaud, and P.-Y. Oudeyer, "Autotelic Agents with Intrinsically Motivated Goal-Conditioned Reinforcement Learning: A Short Survey," *Journal of Machine Learning Research*, vol. 23, pp. 1–41, 2022.

[23] Cognition AI, "Devin 2.0: The AI Software Engineer," Cognition AI Blog, 2025.

[24] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. Narasimhan, "SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering," Princeton NLP, 2024.

[25] "Multi-Agent Curiosity Calibration in Cooperative Exploration," *arXiv preprint*, 2025.

[26] "Human-Like Forgetting Curves in Deep Neural Networks," *arXiv preprint*, 2025.

[27] D. P. McAdams, *The Stories We Live By: Personal Myths and the Making of the Self*. New York: William Morrow, 1993.

[28] "AI-Generated Self-Defining Memories: Narrative Identity in Artificial Agents," *arXiv preprint*, 2025.

[29] G. Loewenstein, "The Psychology of Curiosity: A Review and Reinterpretation," *Psychological Bulletin*, vol. 116, no. 1, pp. 75–98, 1994.

[30] D. Premack and G. Woodruff, "Does the Chimpanzee Have a Theory of Mind?," *Behavioral and Brain Sciences*, vol. 1, no. 4, pp. 515–526, 1978.