

## Comparison: Confucius vs Puffin

Aspect	Confucius Code Agent	Puffin
Purpose	AI software engineer (autonomous agent)	Management layer over Claude Code CLI
Architecture	SDK with orchestrator + extensions	Electron GUI with SAM pattern
Memory	Hierarchical working memory + persistent notes	File-based JSON in .puffin/
Context	Adaptive compression via "Architect" agent	Static config files
Learning	Cross-session learning via note-taking agent	History tracking only
Tool Use	Modular extension system with typed callbacks	Delegates to 3CLI
Transparency	Explicit AX/UX/DX separation	Response viewer

### Key Differences

1. Confucius is a complete agent; Puffin is an orchestration layer  
Confucius replaces the entire agent scaffold (like SWE-Agent). Puffin wraps Claude Code CLI, which means the agent intelligence lives in 3CLI, not Puffin.
2. Confucius separates Agent Experience (AX) from User Experience (UX)  
Users see rich traces with diffs; the agent sees compressed summaries. Puffin currently shows the same information to both user and context.

3. Confucius has persistent cross-session learning The note-taking agent creates hierarchical Markdown notes including "hindsight notes" about failures. These persist across sessions and inform future work.
- 

## What Puffin Can Learn and Implement

### 1. Hierarchical Note-Taking System (High Value)

Add a .puffin/notes/ directory structure: .puffin/notes/ ├── README.md # Summary index ├── projects/ | └─ puffin/ | ├── architecture\_insights.md | └─ recurring\_issues.md └─ shared/ └─ patterns/ └─ electron\_ipc\_gotchas.md - Distill conversation history into reusable knowledge - Track hindsight notes for failures (what went wrong, how it was fixed) - Make notes available as context to 3CLI

### 1. Context Compression Layer (High Value)

Before sending context to 3CLI, implement an "Architect" summarization step: - Summarize long conversation histories while preserving key decisions - Structure context hierarchically (goals, decisions made, open TODOs, errors) - Reduce token usage while maintaining reasoning quality

### 1. AX/UX Separation (Medium Value)

2. User sees: Full streaming output, diffs, verbose logs

3. Agent receives: Compressed summaries, structured memory

4. Add a context preprocessing step before invoking 3CLI

### 5. Extension/Plugin Architecture (Medium Value)

Make Puffin more modular with typed callbacks: // Example extension interface { on\_prompt\_submit: (prompt, context) => modifiedPrompt, on\_response\_received: (response) => processedResponse, on\_session\_complete: (trajectory) => notes } This would allow: - Custom tool integrations - Project-specific behaviors - Observability hooks

### 1. Trace/Observability UI (Medium Value)

Add a "Trace" view showing: - Execution timeline with latency metrics - Token usage per interaction - Tool call hierarchy - Context window utilization

1. Cross-Session Learning Loop (High Value)

Implement a post-session analysis that: 1. Reviews completed tasks 2. Identifies successful strategies 3. Records failure patterns and resolutions 4. Updates project-specific notes

---

#### Suggested Implementation Priority

1. Note-Taking System - Extend .puffin/ with persistent notes that accumulate project knowledge
2. Context Compression - Add summarization before long contexts hit 3CLI
3. Trace UI - Visualize execution for better DX
4. Learning Loop - Analyze sessions to build up knowledge base