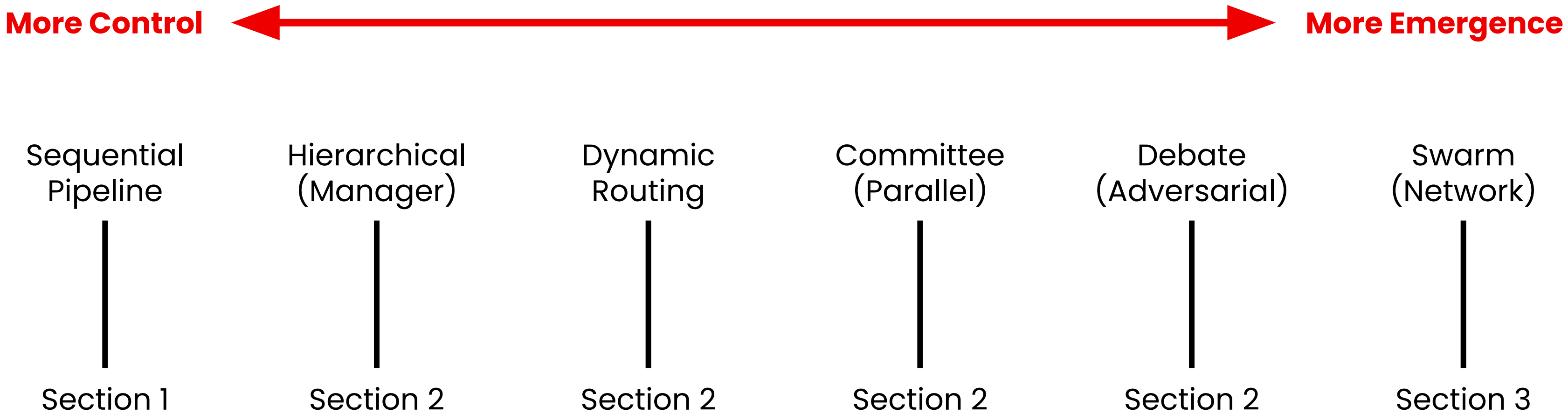
O'REILLY®

Developing with AI Agent Swarms

Agentic software development with claude-code and claude-flow

Section 3: Applying Multi-Agent Patterns

The Control Spectrum



Structured vs. Emergent

Structured (Section 2)	**Emergent (Swarms)**
Predefined workflow	No predefined workflow
Coordinator assigns work	Agents discover work
Explicit handoffs	Shared memory discovery
Predictable behavior	Adaptive behavior

Section 3 Roadmap

- Brief look at emergent coordination (claude-flow)
- Pre-recorded demo of claude-flow
- Decision framework: choosing your architecture
- Best practices, cost, and safety
- Open Q&A

The Agentics Foundation and claude-flow

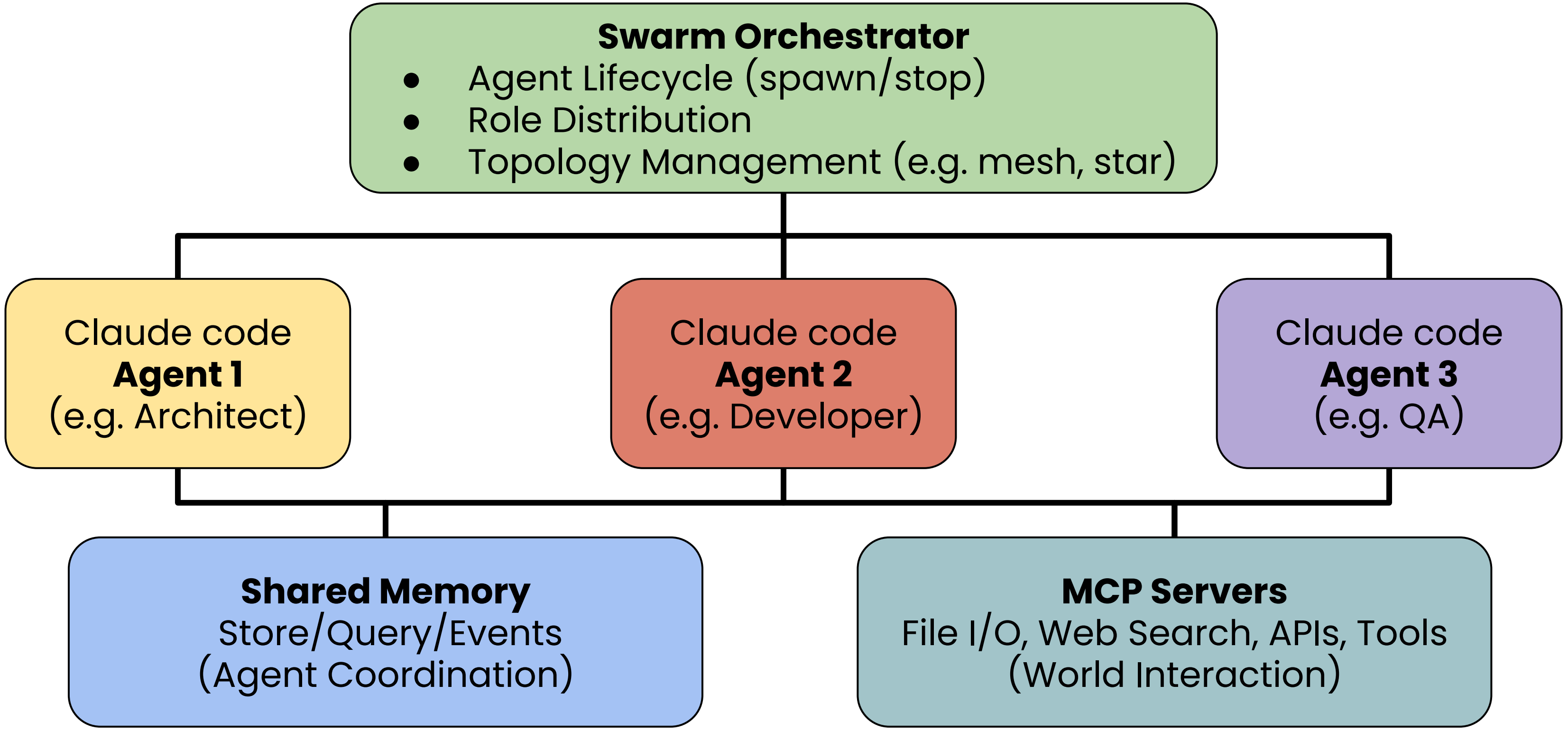
Agentics Foundation

- Community-driven, founded by Reuven Cohen
- Open source swarm orchestration

claude-flow

- Leading swarm platform for Claude
- Multiple agents coordinating via shared memory
- MCP integration for tool access

claude-flow “architecture”



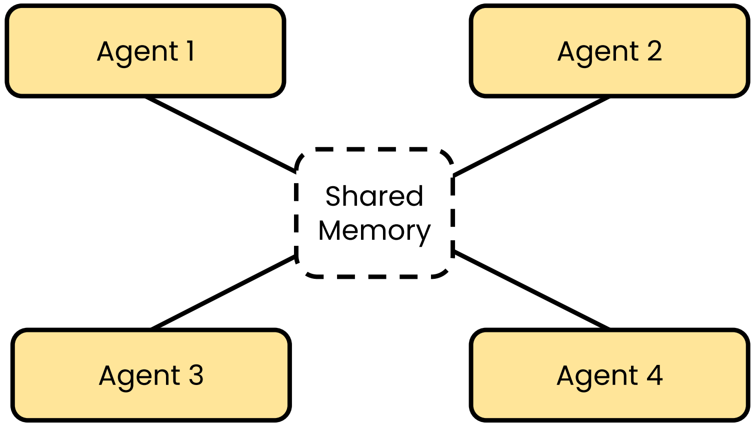
Swarm Orchestrator
● Agent Lifecycle (spawn/stop)
● Role Distribution
● Topology Management (e.g. mesh, star)
Claude code Agent 1 (e.g. Architect)
Claude code Agent 2 (e.g. Developer)
Claude code Agent 3 (e.g. QA)
Shared Memory Store/Query/Events (Agent Coordination)
MCP Servers File I/O, Web Search, APIs, Tools (World Interaction)

The SPARC methodology (for complex projects)

S P A R C

Specification	Pseudocode	Architecture	Refinement	Completion
Define testable requirements	Design algorithms and logic	Define components and systems	Implement via TDD cycles	Integrate, test, deploy, and document
Spec Agent	Design Agent	Architect	Coder Agent	QA/Doc Agents

How Swarms Communicate



1. Agents spawn with specialized capabilities
2. Shared memory stores discoveries, decisions, artifacts
3. Agents query memory to find work and context
4. No central assignment—coordination emerges
5. Results accumulate through collective effort

Demo 3: The Task

The Task:

"Build a simple todo list web application with HTML, CSS, and JavaScript"

Agent	**Role**
Architect	System designer
Frontend	UI developer
Backend	Logic developer
QA	Tester
Documenter	Writer

Uses the basic default methods, **not SPARC**

Demo: The Output Files

Output file(s)

Demo: The Memory Trail

Memory Trail

The Key Difference

Section 2 (Code Review):

"Coordinator, assign Security reviewer to analyze this code"

→ Explicit assignment

Section 3 (Swarm):

"Frontend queries memory, sees architecture, decides to build UI"

→ Discovery, not assignment

The Key Difference

Section 2 (Code Review)	Section 3 (Swarm)
Coordinator assigned reviewers	Agents claimed work
Predefined roles/process	Roles/process emerge organically
Coordinator synthesized at the end	Synthesis emerges from accumulated work
Predictable behavior, auditable	Adaptive, exploratory
Predictable token usage	Unpredictable token usage (usually more than other options)

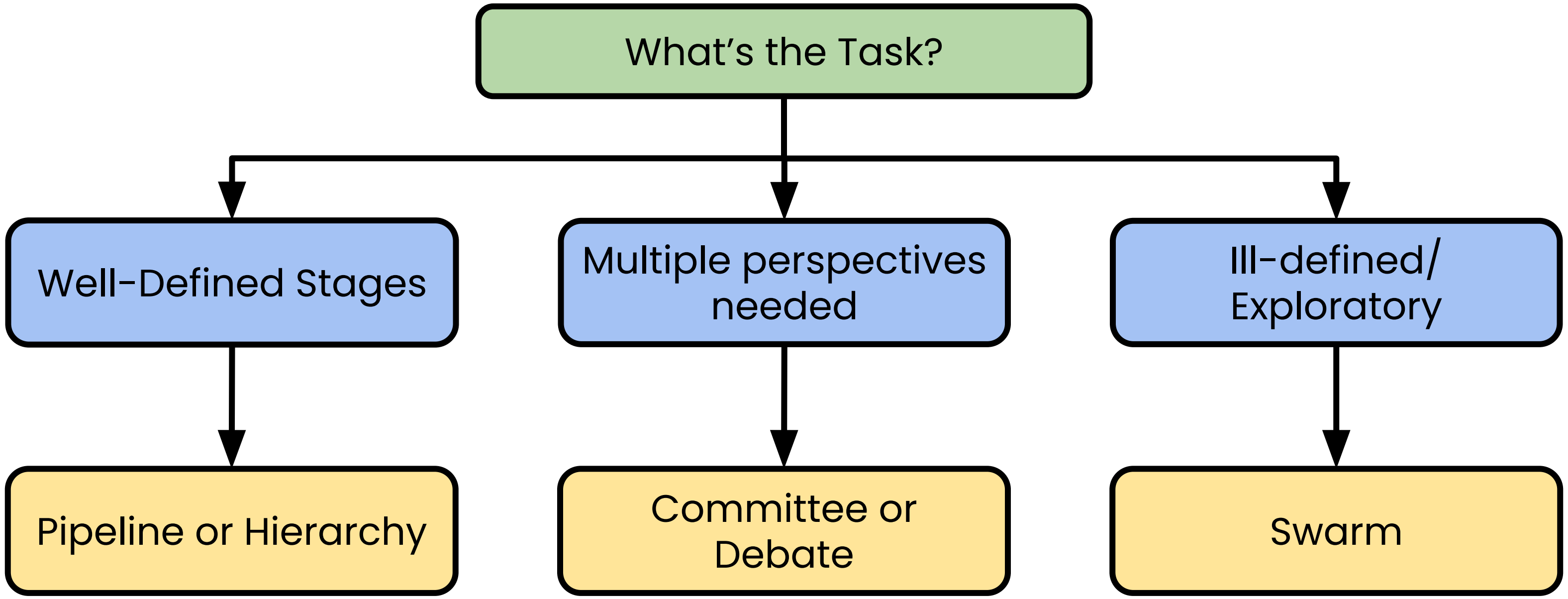
The Central Question

"Given a task, which multi-agent architecture should I use?"

Consider:

- Problem clarity
- Predictability needs
- Time and cost constraints
- Quality requirements

Decision Tree



Pattern Selection Guide

Pattern	Best For	Avoid When
Sequential	Clear stages, transformation chains	Stages need to iterate
Hierarchical	Decomposable tasks with oversight	Workers need to collaborate
Dynamic Routing	Heterogeneous prompts, clear agent expertise	Prompt/query types overlap heavily/are unclear
Committee	Diverse perspectives, reduce bias	Single expert is sufficient, processes are sequential
Debate	High-stakes decisions that require challenge	Time-critical, clear cut decisions
Swarm	Exploratory, ill-defined, adaptive	Need predictability, tight budget

Hybrid Approaches

You don’t have to select just one!

1. Hierarchical with Swarm Workers

    ├── Worker A → Runs mini-swarm for exploration
    ├── Worker B → Runs pipeline for transformation
    └── Worker C → Runs debate for decision

2. Pipeline with Committee Steps

    Research → Committee Review → Write → Committee Edit → Publish

3. Routing to Swarms

    ├── Coding tasks → Coding swarm
    ├── Data tasks → Analysis Pipeline
    └── Worker C → Runs debate for decision

Risk As A Criteria

What would happen if this goes wrong?

Consequence Level	Recommendation
High Consequence	Stick to predictable structure (Hierarchical, Debate)
Low Consequence	Look for more freedom (Swarm, Committee)
Unknown	Start with a swarm, add structure as clarity emerges

Cost As A Criteria

Pattern	Cost Multiplier	Notes
Pipeline	2-5X	Depends on number of stages
Hierarchical	3-10X	Coordinator plus workers
Committee	Nx	Where N is number of members
Debate	2-4X per Round	Compounds with number of rounds
Swarm	5-50X	Highly variable

Cost Controls:

- Set agent limits: --max-agents 4
- Set timeouts: --agent-timeout 300
- Monitor token usage
- Kill idle agents
- Start small, scale up

Safety Rails

Good distribute systems practices still apply!!!

For All Patterns:

- Human approval for destructive actions
- Sandbox file system access
- No production credentials in agent contexts
- Log all significant actions

For Swarms Specifically:

- Circuit breakers (max iterations, max tokens)
- Memory size limits
- Conflict detection for parallel writes
- Regular state snapshots

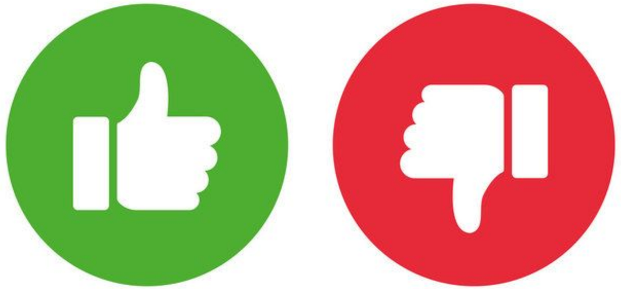
Common Failure Modes

Failure	Symptoms	Prevention
Infinite Loops	Agent count grows, but no progress	Max iteration limits, timeouts
Context Pollution	Agents get confused, produce wrong outputs	Clear handoff formats, output validation
Coordinator Bottleneck	Workers idle, waiting	Increase parallelism or decompose Coordinator work
Duplicate work	Same output from multiple sources	Check shared memory before starting, track task assignment
Conflicting decisions	Inconsistent outputs	Conflict detection, arbitration

Pulse Check

These are a way to quickly check in with attendees. Ask them a simple yes or no question, and the platform will prompt them to press “thumbs up” or “thumbs down”. E.g.

- Does everyone have their Colab environment ready?
- Are you clear on the key architectural differences between our three models?
- Do you feel confident about implementing SLMs in your own projects?



Q&A

Break

O'REILLY®