Multi-Agent Architectures

Week 5: Collaborative AI Systems

PhD Course in Agentic Artificial Intelligence

Learning Objectives

Bloom's Taxonomy Levels

- **Remember**: Define multi-agent systems, coordination, communication
- **Understand**: Explain different agent topologies and roles
- **Apply**: Implement message passing between agents
- **Analyze**: Compare centralized vs. decentralized architectures
- **Evaluate**: Assess trade-offs in multi-agent design
- **Create**: Design a multi-agent system for a complex task

Multi

agent systems enable complex tasks beyond single-agent capabilities.

Why Multi-Agent Systems?

Single Agent Limitations

- Context window constraints
- No specialization (jack of all trades)
- Sequential processing bottleneck

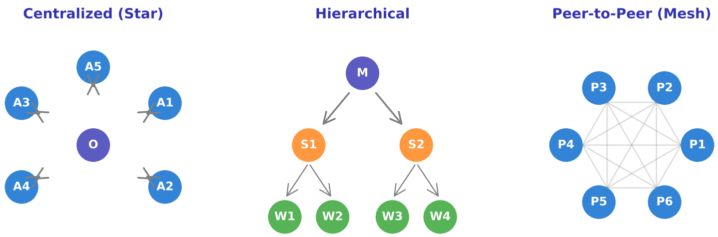
Multi-Agent Benefits

- Role specialization (expert agents)
- Parallel task execution
- Divide and conquer complex problems
- Emergent collective intelligence

Divisi

of labor amplifies capabilities beyond individual agents.

Communication Topologies

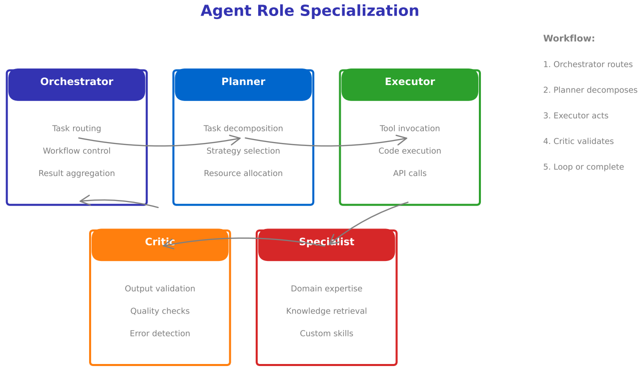


Topo

choice affects coordination overhead and flexibility.

(c) Joerg Osterrieder 2025

Agent Role Specialization



Speci

roles enable division of cognitive labor.

AutoGen Framework

Key Concepts (Wu et al., 2023)

- Conversable agents with defined roles
- Multi-turn message exchange
- Human-in-the-loop (human oversight) integration

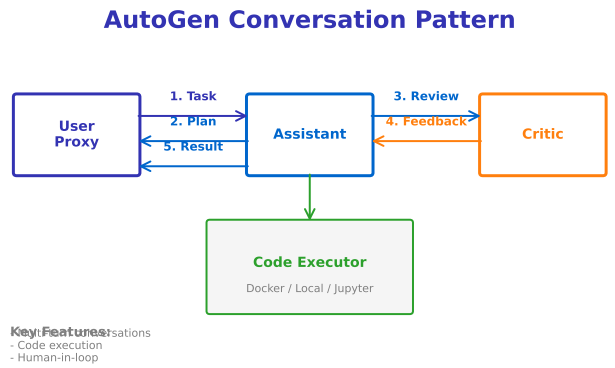
Agent Types

- **AssistantAgent**: LLM-powered responder
- **UserProxyAgent**: Executes code, proxies human
- **GroupChat**: Multi-agent conversation manager

Auto

simplifies multi-agent orchestration with conversation patterns.

AutoGen Conversation Pattern



Auto

enables multi-turn conversations with code execution capability.

MetaGPT: Software Company Simulation

Key Idea (Hong et al., 2023)

- Agents simulate software development roles
- Product Manager, Architect, Engineer, QA
- Structured output protocols between roles

Communication Protocol

- Standardized documents (PRD = Product Reqs, Design Doc)
- Explicit handoff criteria between roles
- Version control for artifacts

Struc

protocols reduce miscommunication in multi-agent systems.

ChatDev: End-to-End Development

Key Idea (Qian et al., 2024)

- Chat-based software development simulation
- Phase-based workflow (Design, Coding, Testing)
- Role-play dialogues between agents

Phases

1. **Design**: CEO + CTO discuss requirements
2. **Coding**: Programmer + Code Reviewer iterate
3. **Testing**: Tester + Programmer fix bugs
4. **Documentation**: Technical writer finalizes

Chatl

generates complete software from natural language specs.

Coordination Mechanisms

Explicit Coordination

- Central orchestrator assigns tasks
- Defined protocols and interfaces
- Clear responsibility boundaries

Emergent Coordination

- Agents negotiate in shared environment
- Consensus through multi-round discussion
- Voting or debate to resolve conflicts

Hybrid Approaches: Structure where needed, flexibility elsewhere

Choo

coordination mechanism based on task predictability.

Message Passing Patterns

Request-Response

- Agent A sends query to Agent B
- B processes and returns result
- Simple, synchronous, easy to trace

Publish-Subscribe (event broadcast)

- Agents subscribe to topics
- Publishers broadcast to all subscribers
- Decoupled, scalable, event-driven

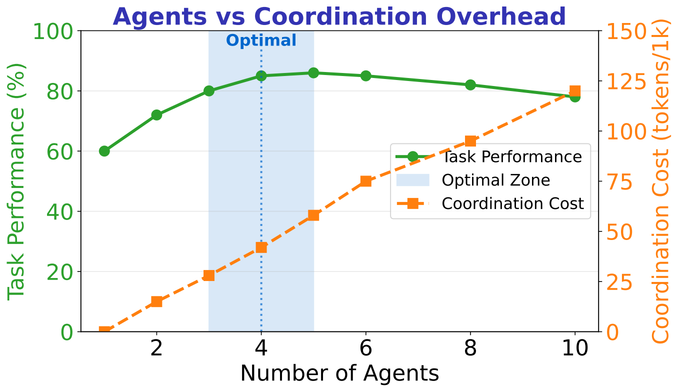
Shared Blackboard

- Common workspace all agents can read/write
- Good for collaborative problem-solving

Patte

choice depends on coupling, latency, and complexity needs.

Coordination Overhead Analysis



Optin

zone: 3-5 agents. Beyond that, coordination costs outweigh benefits.

Challenges in Multi-Agent Systems

Coordination Overhead

- Communication cost increases with agent count
- Synchronization delays
- Context management across agents

Failure Modes

- Cascading errors (one agent fails, chain breaks)
- Infinite loops (agents talking past each other)
- Conflicting actions (race conditions)

Mitigations: Timeouts, circuit breakers (failure isolation), conflict resolution

Multi

agent complexity requires robust error handling.

Design Principles

Best Practices

1. Start with minimal agents, add as needed
2. Define clear role boundaries
3. Use structured output formats
4. Implement graceful degradation
5. Monitor agent interactions

Anti-Patterns

- Too many agents for simple tasks
- Unclear responsibility boundaries
- No termination conditions

Simpl

architectures are easier to debug and maintain.

Required Readings

This Week

- Tran et al. (2025). "Multi-Agent Collaboration Survey." arXiv:2501.06322

Supplementary

- Wu et al. (2023). "AutoGen." arXiv:2308.08155
- Hong et al. (2023). "MetaGPT." arXiv:2308.00352
- Qian et al. (2024). "ChatDev." arXiv:2307.07924

Surve

provides comprehensive overview; frameworks show implementation.

Summary and Key Takeaways

Key Concepts

- **Topologies**: Centralized, hierarchical, peer-to-peer
- **Roles**: Orchestrator, planner, executor, critic
- **Coordination**: Explicit protocols vs. emergent
- **Frameworks**: AutoGen, MetaGPT, ChatDev

Design Decisions

- When to use multi-agent vs. single agent
- Centralized vs. decentralized coordination
- Structured vs. free-form communication

Next Week: Agent Frameworks (LangGraph, CrewAI)

Multi

agent architectures enable complex collaborative intelligence.