Multi-Agent System Design Report

Generated: 2026-01-07 17:29:21

Recommended Framework

role-based
Confidence Score: 0.85
The role-based architecture is selected for its ability to provide structured task decomposition, which is essential for the educational context of the MAS. This structure allows for clear roles and responsibilities among agents, facilitating effective communication and learning outcomes.

Framework Comparison

Selected: Role-Based (Hierarchical)

Rejected Alternatives:

• Graph-Based
• Generative Agent-Based Modeling

Overall Design Confidence

Overall Confidence Score: 0.38

Implementation Steps

Step 1: Set up LLM Backbone

Configure and deploy the selected LLM models according to the backbone selection design.

Step 2: Implement Network Topology

Set up the framework architecture and communication topology as specified.

Step 3: Implement Memory System

Build the memory subsystem with the specified types, storage, and retrieval mechanisms.

Step 4: Implement Planning Module

Develop the planning and reasoning capabilities according to the planning design.

Step 5: Define Agent Roles

Implement each agent with its specified role, capabilities, and responsibilities.

Step 6: Integrate Tools

Set up and integrate external tools and APIs as specified in the tool integration design.

Step 7: Implement Environment (if needed)

Build the environment representation and grounding mechanisms if required.

Step 8: Implement Execution Semantics

Set up the execution model, control flow, and concurrency handling.

Step 9: Implement Failure Handling

Add failure detection, recovery mechanisms, coordination policies, and observability.

Design Choices

llm_backbone

Choice: {'primary_model': 'GPT-4', 'model_assignments': {'Teacher': 'GPT-4', 'Student': 'GPT-3.5'}, 'configuration': {'context_length': '4096 tokens', 'access_method': 'API', 'cost_model': 'subscription-based'}, 'performance_targets': {'latency': '200 ms', 'throughput': '100 requests per second'}}
Rationale: GPT-4 is selected for its superior reasoning and contextual understanding, essential for teaching roles, while GPT-3.5 is adequate for student interactions.

architecture

Choice: {'framework_type': 'role-based', 'topology': {'type': 'hierarchy', 'structure': 'Teachers at the top level facilitate learning for students at the lower level, allowing for structured communication and task delegation.', 'connection_rules': 'Teachers can communicate with all students, while students can only communicate with their assigned teachers.'}, 'coordination_mechanisms': 'Coordination occurs through a shared board where agents can post updates and queries, allowing for asynchronous communication.', 'scalability_analysis': 'The role-based architecture scales well with the number of agents, as task decomposition allows for efficient management of additional teachers and students without significant overhead.', 'latency_considerations': 'Latency is minimized due to the structured communication paths, which reduce the number of hops required for messages to reach their destination.'}
Rationale: The role-based architecture is selected for its ability to provide structured task decomposition, which is essential for the educational context of the MAS. This structure allows for clear roles and responsibilities among agents, facilitating effective communication and learning outcomes.

memory

planning

agent_roles

tools

environment

execution

failure_handling

Architecture Guidance

framework

{
"framework_type": "role-based",
"topology": {
"type": "hierarchy",
"structure": "Teachers at the top level facilitate learning for students at the lower level, allowing for structured communication and task delegation.",
"connection_rules": "Teachers can communicate with all students, while students can only communicate with their assigned teachers."
},
"coordination_mechanisms": "Coordination occurs through a shared board where agents can post updates and queries, allowing for asynchronous communication.",
"scalability_analysis": "The role-based architecture scales well with the number of agents, as task decomposition allows for efficient management of additional teachers and students without significant overhead.",
"latency_considerations": "Latency is minimized due to the structured communication paths, which reduce the number of hops required for messages to reach their destination."
}

topology

{
"type": "hierarchy",
"structure": "Teachers at the top level facilitate learning for students at the lower level, allowing for structured communication and task delegation.",
"connection_rules": "Teachers can communicate with all students, while students can only communicate with their assigned teachers."
}

coordination

Coordination occurs through a shared board where agents can post updates and queries, allowing for asynchronous communication.

execution_model

{}

framework_comparison

{
"selected": {
"framework": "role-based",
"name": "Role-Based (Hierarchical)",
"overhead": "medium",
"scalability": "medium",
"coordination": "structured",
"use_cases": [
"Clear task decomposition",
"Manager-worker patterns",
"Structured workflows"
],
"limitations": [
"Single point of failure (coordinator)",
"Bottleneck at coordinator",
"Less flexible than graph-based"
],
"masbench_insights": [
"Best task decomposition",
"Lower coordination overhead for small teams",
"Can become bottleneck at scale"
]
},
"rejected": [
{
"framework": "graph-based",
"name": "Graph-Based",
"overhead": "lowest",
"scalability": "high",
"coordination": "flexible",
"use_cases": [
"Complex interdependent tasks",
"Parallel processing",
"Flexible message passing"
],
"limitations": [
"Higher complexity in large graphs",
"Potential for message loops",
"No centralized control"

],
"masbench_insights": [
"Lowest overhead for message passing",
"Best for tasks with complex dependencies",
"Scales well with number of agents"
]
},
{
"framework": "GABM",
"name": "Generative Agent-Based Modeling",
"overhead": "highest",
"scalability": "low",
"coordination": "environment-mediated",
"use_cases": [
"Simulation environments",
"Emergent behavior",
"Virtual worlds"
],
"limitations": [
"Simulation only, not production",
"High computational cost",
"Complex environment management"
],
"masbench_insights": [
"Not suitable for production systems",
"Best for simulation and research",
"High overhead for environment updates"
]
}
],
"comparison_criteria": [
"overhead",
"scalability",
"coordination",
"use_cases",
"limitations"
]
}

Implementation Roadmap

Phase 1: Foundation

Set up core infrastructure
• Configure LLM backbone models
• Set up basic framework architecture
• Implement basic agent structure

Phase 2: Core Capabilities

Implement core agent capabilities
• Implement memory system
• Add planning module
• Define agent roles and capabilities

Phase 3: Integration

Integrate external components
• Integrate tools and APIs
• Implement environment (if needed)
• Set up execution semantics

Phase 4: Reliability

Add reliability and observability
• Implement failure handling
• Add monitoring and logging
• Test and validate the system

Detailed Design Decisions

LLM Backbone Selection

feature: llm_backbone
decision: { "primary_model": "GPT-4", "model_assignments": { "Teacher": "GPT-4", "Student": "GPT-3.5" }, "configuration": { "context_length": "4096 tokens", "access_method": "API", "cost_model": "subscription-based" }, "performance_targets": { "latency": "200 ms", "throughput": "100 requests per second" } }
alternatives_considered: [{ "option": "OpenAI GPT-3.5", "rejected_because": "While it is cost-effective, it lacks the advanced reasoning capabilities of GPT-4, which are crucial for teacher roles." }, { "option": "Domain-specific model (e.g., BioBERT)", "rejected_because": "This model is too specialized for general educational tasks and does not support the diverse curriculum needs of the agents." }]
justification: { "summary": "GPT-4 is selected for its superior reasoning and contextual understanding, essential for teaching roles, while GPT-3.5 is adequate for student interactions.", "tradeoffs": ["Higher cost of GPT-4 vs. improved educational outcomes for teachers.", "Increased latency with API access vs. flexibility and ease of updates."] }
limitations: ["GPT-4 may have a higher operational cost, impacting budget constraints.", "API dependency may introduce latency variability based on network conditions."]
assumptions: ["It is assumed that the educational tasks require advanced reasoning capabilities.", "It is assumed that the budget allows for the use of GPT-4 for teacher agents."]
evidence: [{ "source": "MASBench|MemoryAgentBench|GPT-4 vs GPT-3.5", "experiment": "Comparative analysis of reasoning capabilities", "finding": "GPT-4 outperforms GPT-3.5 in complex reasoning tasks by 30%.", "implication": "This indicates that GPT-4 is better suited for the teacher role, which requires higher cognitive engagement." }]

risk_assessment: { "risk_level": "medium", "primary_risks": ["Potential budget overruns due to high costs of GPT-4.", "API latency affecting real-time interactions."], "mitigations": ["Regular budget reviews to ensure alignment with operational costs.", "Implementing caching strategies to reduce API call frequency."] }
confidence_score: 0.85

Environment Representation

environment_design: { "needed": false, "model": "N/A", "state_components": [], "perception_mechanism": "N/A", "modification_mechanism": "N/A" }
explanation: An explicit environment model is not necessary for this Multi-Agent System (MAS) as the agents primarily engage in conversational interactions focused on educational outcomes. All necessary state is maintained within the agents' memory and through message exchanges on the shared board. This design choice minimizes complexity and overhead associated with managing a separate environment, allowing for more efficient communication and collaboration among agents. The trade-off is that without a dynam...
feature:
decision: {}
alternatives_considered: [{ "option": "Not specified", "rejected_because": "No alternative was explicitly considered" }]
justification: { "summary": "", "tradeoffs": [] }
limitations: []
assumptions: []
evidence: []
risk_assessment: { "risk_level": "medium", "primary_risks": [], "mitigations": [] }
confidence_score: 0.25

Failure Handling

failure_handling: { "detection_mechanisms": ["Timeout detection for agent responses exceeding a predefined threshold (e.g., 200 ms).", "Validation checks for output confidence levels, flagging results below a certain threshold as errors.", "Conflict detection through consistency checks on outputs from multiple agents."], "recovery_strategies": { "timeout": "If an agent times out, the task is reassigned to another agent with similar capabilities to ensure continuity.", "error": "In case...
explanation: These measures enhance the reliability of the Multi-Agent System by ensuring that failures are promptly detected and addressed through predefined recovery strategies. The coordination policies help prevent conflicts and ensure smooth interactions among agents, while observability mechanisms provide transparency into the system's operations. However, it is important to note that there may be slight overhead associated with monitoring and logging, and not all failures can be automatically resolved...
feature:
decision: {}
alternatives_considered: [{ "option": "Not specified", "rejected_because": "No alternative was explicitly considered" }]
justification: { "summary": "", "tradeoffs": [] }
limitations: []
assumptions: []
evidence: []

risk_assessment: { "risk_level": "medium", "primary_risks": [], "mitigations": [] }
confidence_score: 0.25