Executive Summary

Multi-Agent LLM Orchestration for High-Quality Incident Response

Philip Drammeh, M.Eng.
philip.drammeh@gmail.com

November 2025

For Engineering Leaders, VPs of Infrastructure, CIOs

Reading Time: 3 minutes

1 The Problem

When production incidents occur, teams face a critical gap: **telemetry arrives in seconds, but actionable understanding takes minutes**. Single-agent AI assistants (like copilots) summarize incidents quickly but generate vague recommendations like "investigate recent changes" that waste operator time.

2 What We Did

We built MyAntFarm.ai—a reproducible framework comparing three approaches across 348 controlled trials:

1. **Manual dashboard analysis** (baseline)
2. **Single-agent AI copilot** (current state-of-the-art)
3. **Multi-agent orchestration** (specialized diagnosis, planning, risk agents)

3 Key Findings

Multi-Agent Systems Are Production-Ready, Single-Agent Are Not

Metric	Single-Agent	Multi-Agent	Impact
Actionable Recommendations	1.7%	100%	**58×**
Action Specificity	0.007	0.557	**80×**
Solution Correctness	0.003	0.417	**140×**
Quality Variance	High	Zero	**SLA-ready**
Speed	41.6s	40.3s	**Parity**

The surprising result: Speed is nearly identical. The value is **deterministic quality**.

What This Means in Practice

Single-Agent (Vague)	Multi-Agent (Specific)
– “Investigate recent changes”	– Rollback auth-service to v2.3.0 using `kubectl rollout undo`
– “Review system metrics”	– Verify database `max_connections=200`
	– Monitor `/api/v1/login` errors for 5min

4 Business Impact

For Your Organization

Current State (Manual + Single-Agent):

- Operators spend 5-15 minutes interpreting vague AI suggestions
- Inconsistent recommendation quality delays MTTR
- No basis for SLA commitments on AI-assisted response

Future State (Multi-Agent):

- **100% actionable recommendations** enable immediate execution
- **Zero variance** supports MTTR SLAs (e.g., “AI recommendations within 60s, 95% confidence”)
- **Reduced cognitive load** on on-call engineers

ROI Estimate

For a team handling **100 incidents/month** with **$200/hour** on-call labor:

Metric	Calculation	Annual Savings
Time saved per incident	5 min (interpretation) → 0 min	—
Labor savings	100 incidents × 5 min × $200/hr	**$20,000/year**
MTTR reduction	10% faster resolution	**$50,000/year**
Total	—	**$70,000/year**

Plus intangibles: Reduced on-call stress, faster learning for junior engineers, fewer escalations.

5 Practical Applications

1. Incident Response Automation

Use Case: Deploy multi-agent system alongside existing runbooks
Implementation: 2-4 weeks pilot with SRE team
Expected Outcome: 50-70% reduction in “what to do next” delays

2. Runbook Generation

Use Case: Generate incident-specific runbooks from historical data
Implementation: RAG integration with postmortem database
Expected Outcome: Context-aware recommendations improving over time

3. Junior Engineer Onboarding

Use Case: Provide high-quality guidance during training
Implementation: Shadow mode during on-call shifts
Expected Outcome: 30% faster ramp-up to independent on-call

4. Compliance & Audit Trails

Use Case: Structured, version-specific remediation logs
Implementation: Export multi-agent outputs to JIRA/ServiceNow
Expected Outcome: Audit-ready incident documentation

6 Limitations & Considerations

Current Limitations

- **Single scenario tested**: Authentication service regression only
- **Small model**: TinyLlama (1B params)—larger models may improve absolute DQ
- **Simulated evaluation**: Not tested in live production incidents
- **No human validation**: DQ scores automated, not validated by SRE experts

Generalization Confidence

- **Architectural advantages** (task specialization, fault isolation) likely persist across scenarios
- **DQ improvement magnitude** may vary by incident type
- **Zero variance property** should hold (derives from deterministic orchestration)

Production Readiness Checklist

Before deploying in your environment:

☐ Validate on 3-5 incident types from your domain

☐ Run human evaluation with 5-10 SRE practitioners

☐ Test with your LLM backend (GPT-4, Claude, Llama 70B)

☐ Integrate with your observability stack (Datadog, Splunk, etc.)

☐ Define rollback criteria (e.g., DQ $< 0.5 \rightarrow$ escalate to human)

7 Next Steps

For Engineering Leaders

1. **Pilot Study** (4 weeks): Run MyAntFarm.ai on 3 recent incidents from your logs
2. **ROI Analysis**: Measure time spent interpreting vague vs. specific recommendations
3. **Integration Planning**: Assess effort to connect multi-agent system to your telemetry

For Researchers

1. **Multi-Scenario Validation**: Test on database, network, storage incidents
2. **Human Studies**: Inter-rater reliability with n=15 SRE experts
3. **Model Scaling**: Evaluate with Llama 3.1 70B, GPT-4, Claude Sonnet

For Practitioners

1. **Clone & Run**: Full reproduction in 30 minutes with Docker
2. **Adapt Scenarios**: Modify incident context to match your environment
3. **Extend Framework**: Add new agent types (security, cost optimization)

8 Implementation Timeline

Phase	**Duration**	**Activities**	**Deliverables**
Proof of Concept	2 weeks	Run on 5 historical incidents	DQ comparison report
Pilot	4 weeks	Live shadow mode with SRE team	Validated accuracy
Integration	8 weeks	Connect to observability stack	Production-ready API
Rollout	4 weeks	Gradual adoption across teams	SLA-backed response

Contact & Resources

- **GitHub**: `https://github.com/Phildram1/myantfarm-ai`
- **Author**: Philip Drammeh, M.Eng. (philip.drammeh@gmail.com)
- **Paper**: Full technical details in LaTeX paper (`paper/main.tex`)
- **Reproducibility**: Complete Docker-based framework included

---

Bottom Line: Multi-agent orchestration is not a performance optimization—it's a **production-readiness** requirement for LLM-based incident response. The 100% actionability rate and zero variance enable SLA commitments impossible with single-agent systems.
Recommended Action: Run a 2-week pilot on your historical incidents to validate findings in your environment.