Specification-Based Fault Localization for LLM-Based Multi-Agent Systems

Mete Aksoy — Supervisors: B. Kulahcioglu Ozkan, A. Panichella, Z. Seyedghorban — EEMCS, TU Delft, 2026

Fault Localization · LLM Multi-Agent Systems · MAST Taxonomy · Dynamic Constraints · GPT-4o

1 The Debugging Problem

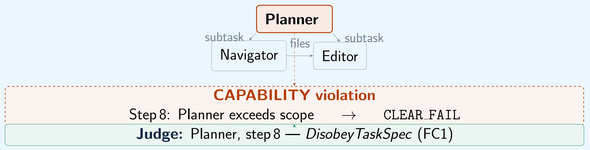
LLM-MAS: multiple AI agents (Planner, Coder, Reviewer...) collaborate on a shared task.

When it fails, we need to know:

- Which agent caused the failure?
- At which step did it occur?
- What category of failure is it?

Why it is hard:

- Errors *propagate* across agents, hiding their origin
- Only an end-to-end pass/fail signal is available
- Execution is non-deterministic



⇒ **Spec-based FL converts an opaque multi-agent failure into an actionable diagnosis — without reading the full trace.**

2 Research Questions

RQ1 — ACCURACY
Can spec-based FL identify failure modes & families in LLM-MAS traces?

RQ2 — METADATA
How does trace metadata availability affect constraint extraction?

RQ3 — EFFECTIVENESS
Which constraint characteristics determine diagnostic effectiveness?

MAST Taxonomy — 3 families, 14 failure modes

FC1 Specification Issues · FC2 Inter-Agent Misalignment · FC3 Task Verification

3 Evaluation Setup

	MAD-Human	HyperAgent-SWE
Traces	18	14
Annotation	3 humans	1 LLM (GPT)
Frameworks	6	HyperAgent
Task	Multi-agent debate	SWE-Bench-Lite
Ground truth	mode & family	mode & family

Baseline: raw LLM judge — no constraints, no violation log
Models: GPT-4o (extraction & judge) · GPT-4o-mini (evaluation)
Runs: 10 independent runs (Full-3c & baseline) on HyperAgent-SWE

4 Six-Stage Pipeline

Stage 1 **Trace Normalization**
Stage 2 **Global Constraint Extraction**
Stage 2b Dynamic Constraint Generation
Stage 3 Constraint Evaluation
Stage 4 Validation Log Construction
Stage 5 **Fault Attribution Judge**
Stage 6 **Evaluation**

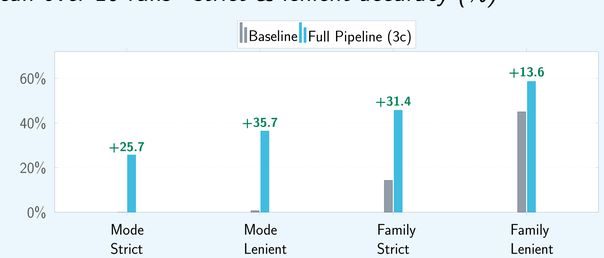
Global constraints — extracted once from system prompts & schemas
Dynamic constraints — GPT-4o generates up to 3 per step, conditioned on trajectory prefix
Validation log — step-indexed `CLEAR_FAIL` records
Judge — reads log, outputs culprit agent · step · failure mode

5 constraint types: CAPABILITY · TEMPORAL · PROTOCOL · RELATIONAL_POST · ANY

⇒ **Dynamic constraints (cyan stages) are the primary diagnostic signal — global extraction fails without system prompts.**

4b HyperAgent-SWE: Baseline vs. Full Pipeline (3c)

Mean over 10 runs · strict & lenient accuracy (%)



5 RQ1: Accuracy Results

Can spec-based FL identify failure modes & families?

MAD-Human (18 traces · strict accuracy)

Mode — Baseline	5.6%
Mode — Full Pipeline (3c)	**33.3%**
Family — Baseline	22.2%
Family — Full Pipeline (3c)	**50.0%**

+27.7 pp mode **+27.8 pp family**

HyperAgent-SWE (14 traces · mean ± std over 10 runs · strict)

Mode — Baseline	0.0% ± 0.0%
Mode — Full Pipeline (3c)	**25.7% ± 6.5%**
Family — Baseline	14.3% ± 0.0%
Family — Full Pipeline (3c)	**45.7% ± 12.0%**

+25.7 pp mode **+31.4 pp family**

⇒ **RQ1: Spec-based FL substantially improves failure mode and family identification on both datasets evaluated.**

6 RQ2: Metadata Availability

How does trace metadata affect constraint extraction?

HyperAgent traces: **no** system prompts, no tool schemas, no role specs
Global extractor: **0 constraints** on 13 of 14 traces
All 513 violations: from **dynamic generation alone**
Dynamic-only still achieves **25.7%** strict mode accuracy (vs 0.0% baseline)

⇒ **RQ2: Metadata-sparse traces rely entirely on dynamic constraints; injecting role schemas as external knowledge would likely improve accuracy further.**

7 RQ3: What Makes Constraints Effective?

Which constraint property drives diagnostic accuracy?

Constraint type → *no discriminating power*
- PROTOCOL dominates both correct & incorrect traces (∼40–45%)
- Distribution nearly identical across both groups

Taxonomy target → *critical signal*
- Constraints naming the correct failure family steer the judge correctly
- Correctly predicted traces: higher concentration of matching targets
- Wrong taxonomy target = misdiagnosis regardless of constraint type

BLIND SPOT — UnawareOfStoppingCond
5 traces with this label — **all misclassified**. Detecting it requires counting repeated failed attempts; no current constraint type captures this pattern.

⇒ **RQ3: The bottleneck is vocabulary (taxonomy targets), not architecture. Cover all failure families in the constraint generator.**

8 Conclusions & Future Work

Main findings:
- Spec-based FL works across **6 heterogeneous frameworks**
- Taxonomy target coverage is the **primary driver** of accuracy
- **3 constraints/step = 5c accuracy** at 67% less cost
- Dynamic constraints enable localization on metadata-sparse traces

Limitations:
- Small datasets (18 + 14 traces)
- No agent/step ground truth — attribution precision unevaluated
- LLM annotation noise in HyperAgent-SWE labels

Future work:
- Dedicated stopping-condition constraint type
- Metadata injection for metadata-sparse systems
- Evaluation on datasets with agent-level ground truth

Evaluated on MAST MAD — 6 frameworks · 14 failure modes · 3 families