INTERACTION PATTERN-BASED FAULT LOCALIZATION IN MULTI-AGENT SYSTEMS

Correlating Agent Execution Sequences with System Failures

Bogdan-Stelian Duminică, EEMCS, TU Delft

TU Delft

Supervisors: Burcu Kulahcioglu Ozkan, Annibale Panichella, Zahra Seyedghorban

1. Introduction

- **Context**: LLM-MAS (LLM-based Multi Agent Systems) handle complex tasks by coordinating specialized autonomous agents via natural language dialogue.
- **Bottleneck**: Debugging LLM-MAS is time-consuming since failures are subtle and non-deterministic, forcing people to manually inspect long execution logs to locate errors.
- **Solution**: This pipeline automatically abstracts raw logs to discover and analyze failure-associated interaction patterns, potentially narrowing the developer search space to reduce debugging time.

2. Research Questions

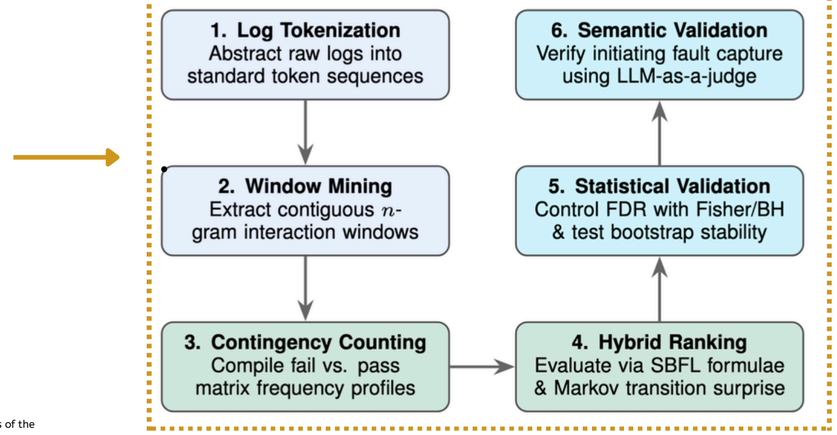
To what extent can interaction pattern-based SBFL and Markov transition analysis identify and prioritize failure-associated interaction patterns in LLM-MAS?

1. How stable and failure-prone are windows ranked by SBFL and Markov surprise across multiple tasks within the same benchmark suite?
2. Which window lengths balance interpretability, stability, and localization usefulness?
3. Can SBFL identify failures in repeated executions of the same task?
4. Do top-ranked windows point to the initiating failures of the runs, not just subsequent symptoms?

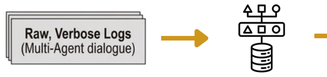
3. Contributions

- **Multi-Framework Tokenizer**: Deterministically maps raw, verbose logs from diverse multi-agent platforms into a sequence of standardized, uniform actions.
- **Hybrid Fault Localization**: Combines coverage-based SBFL formulas with conditional Markov Chain Surprise to isolate both frequent failure-linked windows and anomalous conversational transitions, backed by statistical validation.
- **Semantic Validation**: Use an LLM-as-a-judge approach to verify that top-ranked windows capture actual initiating faults rather than downstream system symptoms.

4. Methodology



5. Log Abstraction & Localization Metrics

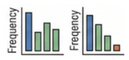


Abstraction Pipeline

Uniform Alphabet (8 tokens)

Orchestration & Execution Tokens	Validation & Outcome Tokens
REQUEST (Task Specs)	**VERIFY_PASS** (Validation Check)
INFORM (Context Delivery)	**VERIFY_FAIL** (Critic Challenge)
ACT (Tool Use/Code)	**TERMINATE** Task / Execution End
PLAN (Role Handoffs)	**ERROR** (Runtime Crash)

Coverage Frequency (Ochiai)



$$\text{Ochiai}(g) = \frac{e_f(g)}{\sqrt{(e_f(g) + n_f(g))(e_f(g) + e_p(g))}}$$

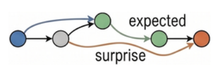
Flags interaction windows highly frequent in failures but rare in passes.

Reciprocal Rank Fusion (RRF)

$$\text{RRF}_{\text{formula}}(g) = \sum_{m \in \{\text{Ochiai, Tarantula, } D^*\text{, Jaccard, OP2}\}} \frac{1}{k + r_m(g)}$$

Combines multiple SBFL formulae for structural consistency.

Sequence Probability (Markov Surprise)



$$\Delta_{\text{surprise}}(g) = \mathbb{E}_{\text{fail}}[S(s_i) \mid g \in s_i] - \mathbb{E}_{\text{pass}}[S(s_i) \mid g \in s_i]$$

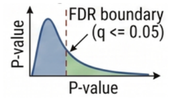
Identifies anomalous transitions relative to a pass-only baseline model.

Fisher's Exact Test

	Failure	Pass
Present	a	b
Absent	c	d

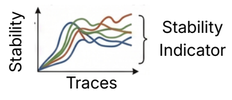
Evaluates statistical correlation. Filters random co-occurrences.

Benjamini-Hochberg (BH) Adjustment



Controls False Discovery Rate. Corrects multiple testing.

Nonparametric Resampling (Bootstrap)



Confirms top patterns are structurally stable over trace variations.

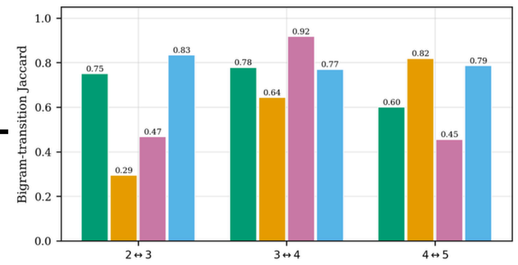
6. Results & Evaluation

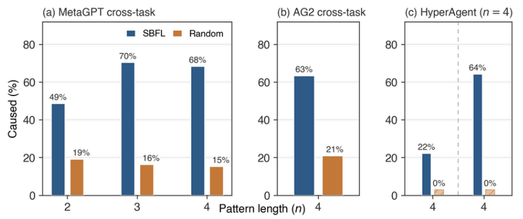
Fw.	Rk	Pattern	Och.	Δ	Tier
MetaGPT	1	REQ:PLAN:ACT:VF	0.74	0.11	bh_sig.
	2	VF:INF:VF:INF	0.73	0.10	bh_sig.
	3	PLAN:ACT:VF:ACT	0.70	0.06	bh_sig.
ChatDev	1	ACT:INF:ACT:TERM	0.70	0.10	descript.
	2	ACT:VP:PLAN:ACT	0.64	0.05	descript.
	3	VP:PLAN:ACT:INF	0.62	0.04	bh_sig.
HyperAgent	1	PLAN:ACT:TERM:ACT	0.62	0.05	descript.
	2	TERM:PLAN:ACT:PLAN	0.58	0.17	descript.
	3	ACT:PLAN:ACT:PLAN	0.54	0.14	descript.
AG2	1	REQ:PLAN:ACT:INF	0.65	0.11	descript.
	2	PLAN:ACT:INF:TERM	0.56	0.15	descript.
	3	ACT:INF:TERM:INF	0.50	0.17	descript.

Abbrev.: REQ = REQUEST; INF = INFORM; TERM = TERMINATE; VP = VERIFY_PASS; VF = VERIFY_FAIL. Ranks by Ochiai (SBFL).

Cross-Task Patterns: With a 100% top-3 pattern overlap across all four frameworks, SBFL and Markov metrics capture robust, architectural vulnerabilities rather than metric-specific noise.

Window Stability: The n = 3 - 4 granularity transition reaches a peak mean Jaccard overlap of ~0.78, showing that this window length optimally balances semantic interpretability with data sparsity.





Semantic Validation: Prioritized rank-1 windows contain initiating faults in 63% to 70% of failing runs versus just 15% to 21% for random controls , meaning that pattern-guided localization considerably narrows the developer search space to mitigate the time-consuming process of undirected log inspection.

7. Conclusions & Future Work

This research demonstrates that modeling verbose multi-agent logs as sequential execution paths over a uniform alphabet isolates systemic architectural vulnerabilities without relying on handcrafted rules . The prioritized execution windows significantly outperform random controls by pinpointing initiating failures across multiple frameworks, mitigating the bottleneck of undirected, manual log inspection.

Future Work

- Transitioning from offline analysis to live operations to enable real-time pattern matching and potential automated self-repair loops mid-run .
- Conducting empirical user studies to formally quantify the exact debugging time saved compared to standard log inspection.
- Refining the log parser and expanding to other LLM-MAS.

References

[1] R. Abreu, P. Zoeteweij, and A. J. C. van Gemund. On the accuracy of spectrum-based fault localization. In Proceedings of the Testing: Academic and Industrial Conference Practice and Research Techniques - MUTATION (TAICPART-MUTATION), 2007.
[2] M. Cemri et al. Why do multi-agent LLM systems fail? In Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track, 2025.
[3] S. Wang et al. Bugram: Bug detection with n-gram language models. In Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (ASE), 2016
[4] Y. Ge et al. Who is introducing the failure? automatically attributing failures of multi-agent systems via spectrum analysis. arXiv preprint arXiv:2509.13782, 2025