Collaborative multi-agent language models for software development

Moises Diaz Malagón, José Mtanous Treviño, Mazal Bethany, Anna Karen Gárate Escamilla, Juan Arturo Nolazco Flores, Paul Rad

Master of Applied Artificial Intelligence / Tecnológico de Monterrey, The University of Texas at San Antonio, School of Data Science

Tecnológico de Monterrey

The University of Texas at San Antonio
UTSA School of Data Science

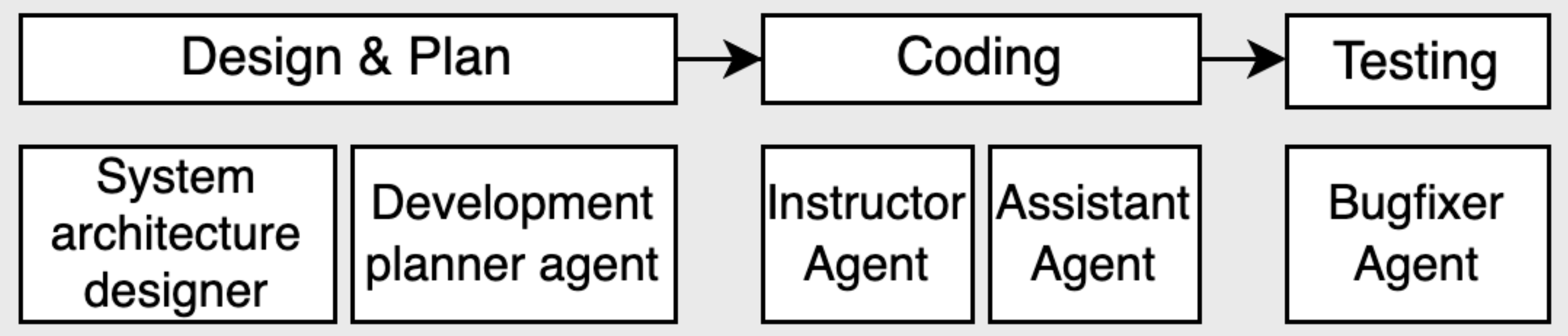
Abstract & Project Goal

Software development requires cooperation among people with diverse skills and typically includes design, coding, and testing stages. Past deep learning techniques often needed custom designs and were ineffective for software development's non-deterministic behavior. Multi-agent LLM-powered systems communicate through multi-turn dialogues. Our goal is to propose a new method for using LLMs in software (backend) development, leveraging agent-based approaches and advanced LLM prompting techniques, while exploring state-of-the-art evaluation methods for software generation.

Background

LLMs excel at handling natural language, aiding in system design and communication. Techniques like chain-of-thought help LLMs solve problems. Role-playing lets LLMs adopt roles without fine-tuning. Autonomous agents enhance LLMs with function calling and memory, enabling them to reflect on actions and context. Software development, a complex task, requires multiple LLM-powered agents with specific roles for different phases. Evaluating generated software is difficult. Typical metrics like pass@k are insufficient. The ChatDev paper suggests completeness, executability, and consistency as metrics, integrating them into a single quality score.

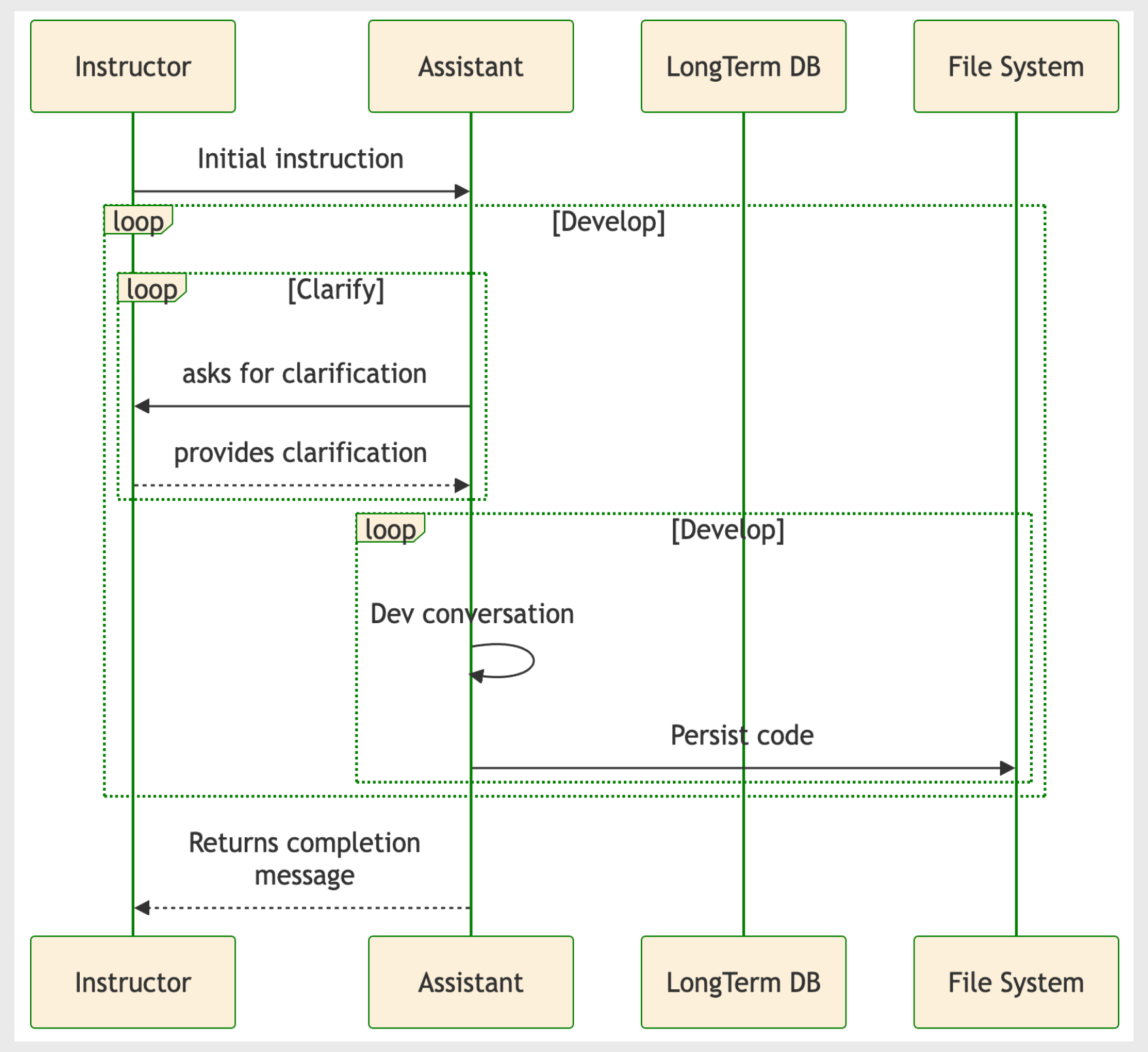
Methods



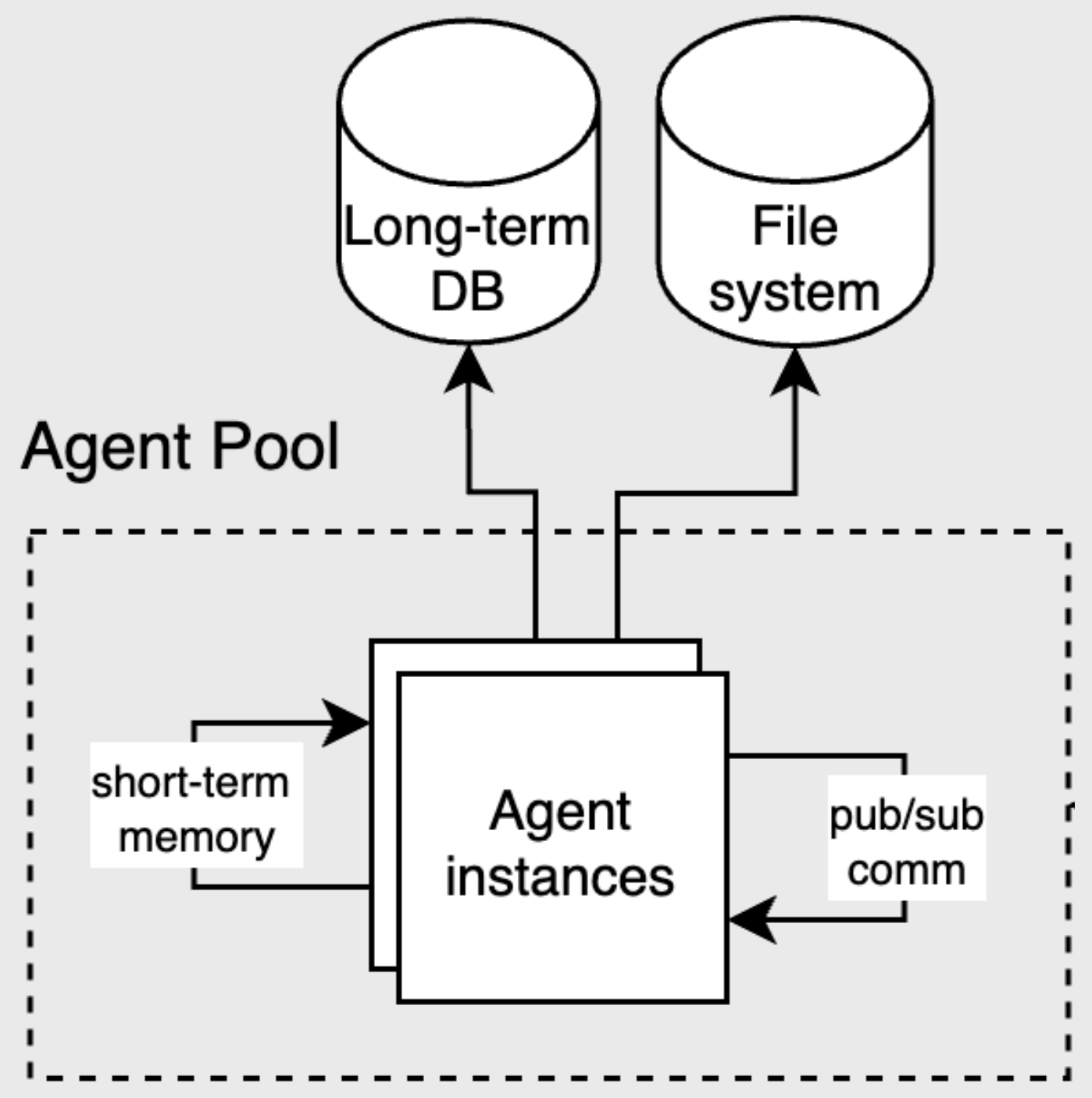
High-level agentic process

ChatDev consists of multiple agents for design, coding and testing, cooperating in natural language.

- Agents perform multi-turn communication for development between different roles.
- Utilizes *communicative dehallucination* mechanism which consists on two agents one instructor and one assistant in which the coding assistant asks for clarifications before giving a final response.



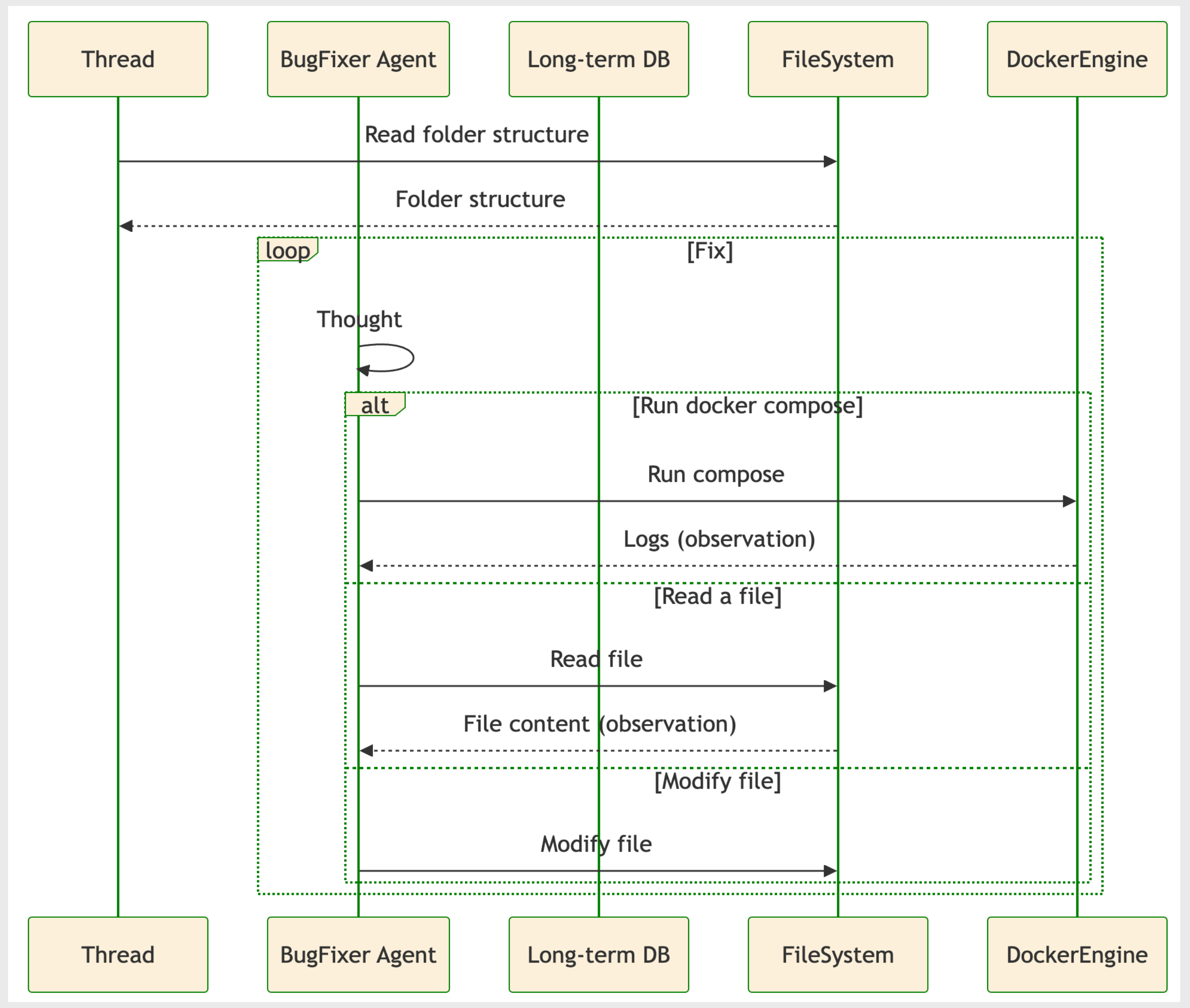
Coding System



High-level architecture

Evaluations metrics proposed by chat dev are: Executability: evaluates software's operation in a compilation environment. Consistency: assesses how the generated code matches the original requirement using similarity on embeddings. Completeness: quantifies the percentage of software without placeholder code. Quality: a comprehensive metric combining the previous factors.

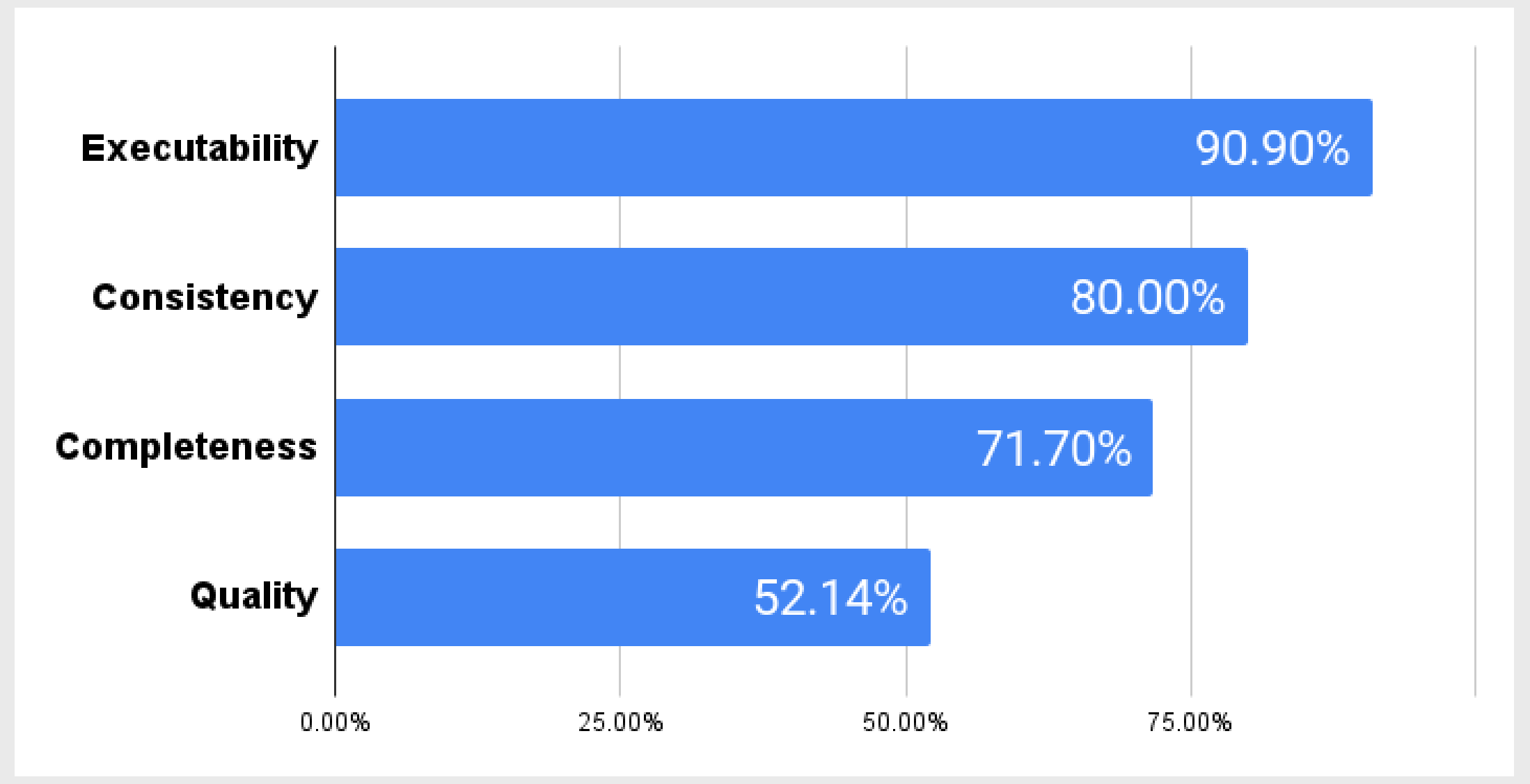
Key components of our proposed system are: System designer and planner, coding system, bugfixer system. Tehcniques used include multi-turn collaborative agents, communicative dehallucination, repeated instruction mechanism, ReAct agents, chain-of-thought and few-shot learning.



Bugfixer System

Results

870 ChatDev test prompts were used to generate the 4 metrics, obtaining the following results:



The new system proposal was evaluated for completeness and executability in developing a social network. More testing is required for representative metrics.

Conclusions

- Multi-agent techniques as well as advanced prompting are key to the solution of complex problems with LLMs.
- Information extractors, dehallucination and repeated instruction mechanism greatly help for more deterministic workflows using LLMs.
- More research is required for the definition of useful metrics for generated code.

Future Work

- Extensive work on testing and metrics is required.
- Build a metric to measure code vulnerability
- Coordinator agent for spinning up agents.
- Functions to consult web pages/documentation.

References

Qian, C., Cong, X., Yang, C., Chen, W., Su, Y., Xu, J., Liu, Z. and Sun, M., 2023. Communicative agents for software development. arXiv preprint arXiv:2307.07924.