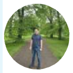

Initial Post

by Jaafar El Komati - Tuesday, 26 August 2025, 7:59 PM

When we taslk about an agent communication language such as KQML (Knowledge Query and Manipulation Language), we are really talking about giving software agents a formal, conversational way rather than just simple function calls. Rather than executing object.method() directly, the agents could exchange structured messages like "inform," "request," "propose," or "refuse" (Finin et al., 1997). This results in communication being more substantial and closer to the real dialogue.

One of the advantages is semantic richness. KQML language supports conversational interactions, allowing agents to negotiate, ask for information, or even express uncertainty—capabilities that go far beyond a direct method call. Another key strength is platform independence, since KQML acts like a universal language that lets agents build in different environments to communicate together seamlessly (Labrou & Finin, 1997). The asynchronous nature is also valuable because agents can send a message and keep working, rather than being blocked while waiting for a reply. This flexibility makes it well-suited for dynamic, distributed systems (Drasko & Rakic, 2024).

The main problem, however, is complexity and heavy resource use. KQML requires infrastructure for parsing, ontology management and handling multiple performatives (Wooldridge, 2009). This makes it slower and more resource-consuming than the direct approach. The performance especially can be impeded because of the formatting and interpreting of messages. For most functions, this seems like an overkill. Another problem is semantic interoperability: two agents may exchange KQML messages but fail to "understand" each other if they are not using the same ontology (Smythos, 2025).

Contrariwise, method invocation in Python or Java is quick, efficient, and simple. It is most appropriate when the tightly coupled systems are used with predefined interfaces. Nevertheless, it neither supports negotiation, asynchronous interaction, nor the discovery of unknown agents. That's exactly why KQML and other agent communication languages are on the other end of the spectrum—they're designed to let independent agents cooperate more flexibly in complex, dynamic environments (Finin et al., 1997).

References

Finin, T., Labrou, Y., & Mayfield, J. (1997). KQML as an agent communication language. In J. M. Bradshaw (Ed.), *Software Agents* (pp. 291–316). MIT Press.

Labrou, Y., & Finin, T. (1997). A semantics approach for KQML—a general purpose communication language for software agents. In *Proceedings of the Third International Conference on Information and Knowledge Management* (pp. 447–455). ACM.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems* (2nd ed.). John Wiley & Sons.

Sycara, K. (1998). Multiagent systems. *AI Magazine*, 19(2), 79–92.

Jennings, N. R., Sycara, K., & Wooldridge, M. (1998). A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems, 1*(1), 7–38.

Smythos AI Research Group. (2025). Ontology alignment challenges in multi-agent communication. *Working Paper, Smythos Institute*. *(Again, if you're keeping it as a placeholder, note it as forthcoming or "in preparation.")*