

The Notion of an *Agent* as a Practical Software Engineering Abstraction (Position Statement)

Timotheus Kampik, Umeå University, tkampik@cs.umu.se

In today's interconnected information technology ecosystems, software engineering abstractions that are designed to encapsulate *autonomous* behavior are of increasing relevance. In the academic Artificial Intelligence community, the notion of an *agent* is the most prominent concept in this regard, and an active sub-community works on agent-oriented software engineering abstractions, in particular on *Agent-Oriented Programming* (AOP) languages and frameworks. While existing AOP variants excel at facilitating scientific exploration at the intersection of academic programming and knowledge representation & reasoning, making the case for agent-orientation in practical software engineering is an open problem. To address this problem, *minimally viable* agent-oriented abstractions need to be designed in the context of mainstream programming languages and frameworks. These abstractions should:

1. Provide a clear and intuitive value proposition to a (perhaps somewhat intellectually curious) practicing software engineer -- even one without a Computer Science degree;
2. Not add any non-essential overhead that steepens the learning curve and hampers adoption;
3. Come with clear design principles with regard to agent internals (reasoning cycle) and agent interface design (interaction and discoverability);
4. Allow for a seamless integration into continuous integration and DevOps practices, both conceptually and from a tooling perspective.

Potential learnings can be drawn from the relatively recent spread of functional programming flavors to mainstream and originally predominantly object-oriented programming languages and frameworks, a high-profile example being React.js in the context of JavaScript (for better or for worse).