Agent-Based Software Engineering

Long Abstract

Charles Petrie
Stanford Networking Research Center
Stanford, CA 94305-2232
<petrie@stanford.edu>
http://snrc.stanford.edu/~petrie

It has previously been claimed that agent technologies facilitate software development by virtue of their high-level abstractions for interactions. We address a more specific characterization and utility. We believe that it is important to distinguish agent technologies from other software technologies by virtue of a set of unique software characteristics. This is in contrast to much in the literature that concentrates on high-level characteristics that could be implemented with a variety of software techniques.

Agent-based software engineering (ABSE), for at least an important class of agents and applications, can be characterized by both model and inner/outer language components. Our experience in developing applications based on long-term asynchronous exchange of agent messages, similar to typical email usage, leads us to believe these unique characteristics facilitate useful software development practices. The utility derives from a stratification of change among the components, ease of collaborative change and debugging even during runtime due to asynchronous text parsing-based message exchange, and reuse of the outer language as well as generic agents as a programming environment.

1. Agent Development Characteristics

Jennings and Wooldridge have described Agent-Oriented Software Engineering (AOSE) (Wooldridge and Jennings 99) (Jennings and Wooldridge 00). AOSE effectiveness claims are based upon three strategies for addressing complex systems: decomposition, abstraction, and organization and that the "agent-oriented mindset" gives one an advantage in using these strategies. That agents have objectives gives one a clear way to decompose the problem into agents. That agent systems work largely by emergent behavior and handle errors gracefully reduces the need for detailed specifications, since not all interactions need be specified in advance, and allows more abstraction to be used in system building. And finally, agent systems are naturally hierarchical organizations themselves. Indeed, it is important to note that agent identity is a fundamental component of all agent languages and methodologies for interactions.

We agree with these points. In particular, the idea of engineering a system so that the correct emergent behavior results is the most critical idea in agent software engineering. We further agree with the notions introduced in (Huhns 00) that one aspect of emergent behavior is having software modules being able to model themselves and other modules,
which leads to modules that are able to attempt different methods of accomplishing a task based upon runtime data and these models, which in turn leads to flexibility and robustness without the necessity of the programmer having correctly considering every possibility. The fundamental idea here is that the programmer focuses on the types of interactions possible without specifying all possibilities in advance and one technique for doing this is modeling interaction behaviors. These notions are important and fundamental. This paper is also important because it points out that software engineering seems to have reached a plateau of results and has been stuck there for some time. Thus the agents approach is potentially very important.

However, we find the AOSE explanations lacking in the detail that would allow a software engineer to decide easily whether an AOSE approach was even being used or not. We describe Agent-Based Software Engineering (ABSE) here as a refinement of some aspects of AOSE, based upon our practical experience in agent building. We do not, though, attempt to describe how to practice ABSE. Our goal is rather show that ABSE could be distinguished objectively from other software techniques, which has not previously been done.

This is a topic that is frequently addressed with respect to languages. In fact, in the abstract of a recent paper on agent languages, M. Wooldridge said his intent was to develop "a" semantics where conformance (or otherwise) to the semantics could be determined by the independent observer" (Wooldridge 98). Whether something is an agent or not, and whether a software engineering technology is agent-based or not, should also be verifiable by independent observers.

Further, we attempt to persuade the user that ABSE could be useful in practice and why, apart from the general potential of emergent behavior. We emphasize some particular aspects of agent models and languages that distinguish agent system development from other software technologies and make it useful. Our intent is to help bridge the gap between agent technology and software engineering.


See also ftp://ftp.elec.qmw.ac.uk/pub/isag/distributed-ai/publications/agt-handbook.pdf

