# Intelligent Software Agents

**Alexa Heucke**
*Munich University of Applied Sciences, Germany*

**Georg Peters**
*Munich University of Applied Sciences, Germany*

**Roger Tagg**
*University of South Australia, Australia*

## INTRODUCTION

An agent, in traditional use of the word, is a person that acts on behalf of another person or group of persons. In software, the term *agent* is broadly used to describe software that carries out a specialist range of tasks, on behalf of either a human user or other pieces of software. Such a concept is not new in computing. Similar things have been said about subroutines, re-usable objects, components and Web services. So what makes agents more than just another computer technology buzzword and research fashion?

## BACKGROUND

The idea of intelligent agents in computing goes back several decades. Foner (1993, p. 1) dates the first research on software agents to the late 1950s and early 1960s. However, with the breakthrough of the Internet, intelligent agents have become more intensively researched since the early 1990s. In spite of this long heritage, the uptake of these ideas in practice has been patchy, although the perceived situation may be partly clouded by commercial secrecy considerations. Even today, the many different notions of the term *software agent* suggest that the computing profession has not yet reached a generally accepted understanding of exactly what an agent is.

## DEFINITIONS AND CLASSIFICATIONS

According to Jennings et al. (1998, p. 8), "an agent is a computer system, *situated* in some environment that is capable of *flexible autonomous* action in order to meet its design objectives".

- S*ituated:* the agent interacts directly with its environment. It receives inputs from the environment and it performs activities with external effects.
- *Autonomous:* the agent is in charge of its own internal status and actions. Thus, it can perform without explicit interference of a user or other agents.
- *Flexible:* the agent is responsive to its environment and, at the same time, proactive. The agent should show social behaviour and should feature the ability to interact with external entities.

Furthermore, many agents show intelligence, in that they "carry out some set of operations on behalf of a user or another program with some degree of independence or autonomy, and in doing so, employ some knowledge or representation of the user's goals or desire" (Gilbert et al., 1995, p. 2f).

The research literature discusses many different types of agents, carrying out all sorts of functions with what can be termed primary and secondary characteristics. Primary characteristics include autonomy, cooperation and learning, while secondary characteristics include aspects like multi-functionality, goodwill or trustiness.

A typology of software agents has been proposed by Nwana (1996, pp. 7-38):

- Collaborative agents feature a high degree of cooperation and autonomy. They are determined by the idea of distributed artificial intelligence and by the concept of task sharing, cooperation and negotiation between agents.
- Interface agents focus on the characteristics of learning and autonomy. By collaborating with the user and by sharing knowledge with other agents they learn a user's behaviour and are trained to take the initiative to act appropriately.
- Mobile agents are not static but have the ability to travel. This entails non-functional benefits such as freeing local resources, showing more flexibility and enabling an asynchronous work scenario.

- Information or Internet agents emphasise managing enormous amounts of information. Their main task is to know where to search for information, how to retrieve it and how to aggregate it.
- Reactive agents are showing a stimulus-response manner as opposed to acting deliberatively. Since they are based in the physical world and only react to present changes, their behaviour is not predetermined.
- Hybrid agents comprise more than one agent philosophy and benefit from the combination of different architectures.

Wooldridge and Jennings (1995, pp. 24-30) offer a two-way classification, based on contrasting approaches to building agents. They distinguish the following representative architectures:

- Deliberative agent architecture. This classical agent architecture consists of one definite, symbolic world model with all decisions being made on the basis of logical reasoning. Challenges of this approach are the translation of the real world into an accurate model and the establishment of an efficient reasoning.
- Reactive agent architecture. In contrast to the deliberative agent architecture this alternative approach is lacking an explicit and symbolic model of the world as well as extensive reasoning.

Wooldridge and Jennings also allow for hybrid agent architectures that are built as a hierarchy of deliberative and reactive agent architecture layers.

## DISCUSSION

Four aspects are of particular interest when trying to understand how agents work and could be successfully employed in applications and environments. Discussed next are: agent knowledge, agent applications, agent standards and multi-agent systems.

### Agent Knowledge

To operate autonomously, any software agent must build up a collection of knowledge, typically data and rules that enable it to serve the human or client software it is acting for. According to Maes (1994, p. 2f), an agent's knowledge base should be built up gradually by learning from users and other agents. The key issues are competence and trust. To be competent, the agent must have a knowledge base that is comprehensive and flexible enough to adapt to the user's profile. For an agent to be trusted, a human user must feel comfortable when accepting help from the agent or when delegating tasks to it. Generally, an agent can only learn from its user and other agents if their actions show an iterative pattern. Maes suggests four different ways of training an agent to build up competence: observation and imitation of the user's habits, user feedback, training by examples and training by other agents.

However, Nwana and Ndumu (1999, p. 10) have criticized Maes' approach, claiming that an agent would not only need to know all peculiarities of the deployed operating system but also must understand all tasks its user is engaged in. Furthermore, the agent would need to be capable of gathering the user's intent at any time, thus continuously modelling its user. Nwana and Ndumu identify four main competences for an agent: domain knowledge about the application, a model of its user, strategies for assistance and a catalogue of typical problems that users face in the environment.

### Agent Applications

Software agents can be employed in many fields of information technology. One role for agents is to act as an assistant or helper to an individual user who is working with a complex computer system or physical equipment. Examples are:

- Information agents (Davies et al., 1996, pp. 105-108) that help a human researcher in finding the most relevant material, from many sources and possibly through different search engines.
- Decision support agents that help a user assess alternative courses of action; functions include filtering and summarisation of data, optimising algorithms, heuristics and so forth.
- E-mail agents (Maes, 1994, p. 5f), which filter spam, allocate incoming mail to folders, and work out addresses to which outgoing mail should be sent.
- Buying and selling agents, which assist a user in finding good deals in Internet marketplaces or bidding agents, which assist participants in auctions. These agents have characteristics of information agents as well as of decision support agents.

A second group of applications is where the agent acts as a coordinator of activities, or "virtual manager". Any workflow management system could qualify for this category. Other examples include meeting scheduling agents (Kozierok, 1993, p. 5), and dynamic scheduling agents that are able to re-allocate resources to meet the goals of a business process (Lander et al., 1999, p. 1ff).