Agent Frameworks

Reinier Zwitserloot
Delft University of Technology, The Netherlands

Maja Pantic
Delft University of Technology, The Netherlands

INTRODUCTION

Software agent technology generally is defined as the area that deals with writing software in such a way that it is autonomous. In this definition, the word *autonomous* indicates that the software has the ability to react to changes in its environment in a way that it can continue to perform its intended job. Specifically, changes in its input channels, its output channels, and the changes in or the addition or removal of other agent software should cause the agent to change its own behavior in order to function properly in the new environment. In other words, the term *software agent* refers to the fact that a certain piece of software likely will be able to run more reliably without user intervention in a changing environment compared to similar software designed without the software agent paradigm in mind. This definition is quite broad; for example, an alarm clock that automatically accounts for daylight savings time could be said to be autonomous in this property; a change in its environment (namely, the arrival of daylight savings time) causes the software running the clock to adjust the time it displays to the user by one hour, preserving, in the process, its intended function—displaying the current time. A more detailed description of agent technology is available from Russel and Norvig (2003).

The autonomous nature of software agents makes them the perfect candidate for operating in an environment where the available software continually changes. Generally, this type of technology is referred to as multi-agent systems (MAS). In the case of MAS, the various agents running on the system adapt and account for the other agents available in the system that are relevant to its own operation in some way. For example, MAS-aware agents often are envisioned to have a way of negotiating for the use of a scarce resource with other agents.

An obvious start for developing MAS is to decide on a common set of rules to which each agent will adhere, and on an appropriate communication standard. These requirements force the need for an underlying piece of software called an agent framework. This framework hosts the agents, is responsible for ensuring that the agents keep to the rules that apply to the situation, and streamlines communication between the agents themselves and external sensors and actuators (in essence, input and output, respectively). This paper will go into more detail regarding the advantages of MAS and agent frameworks, the nature and properties of agent frameworks, a selection of frameworks available at the moment, and attempts to draw some conclusions and best practices by analyzing the currently available framework technology.

BACKGROUND: RESEARCH MOTIVATIONS

An agent framework and its use as a base for MAS technology already has been successfully used as the underlying technology for most teams participating in the robot soccer tournament (Tambe, 1998). The robotic soccer tournament requires that all participating robot teams operate entirely under their own control without any intervention by their owners. The general idea of independent autonomous robots working together to perform a common task can be useful in many critical situations. For example, in rescue situations, a swarm of heterogeneous (not the same hardware and/or software) agents controlling various pieces of hardware fitted onto robots potentially can seek out and even rescue

people trapped in a collapsed building. The ideal strived for in this situation is a system whereby a number of locator robots, equipped with a legged transport system to climb across any obstacle and sporting various location equipment such as audio and heat sensors, will rapidly traverse the entirety of the disaster area, creating a picture of potential rescue sites. These, in turn, serve as the basis for heavy tracked robots equipped with digging equipment, which work together with structure scanning robots that help the digging robots decide which pieces to move in order to minimize the chances of accidentally causing a further collapse in an unstable pile of rubble. Equipment breaking down or becoming disabled, for example, due to getting crushed under an avalanche of falling rubble, or falling down in such a way that it can't get up, are not a problem when such a rescue system is designed with MAS concepts in mind; as all agents (each agent powering a single robot in the system) are independent and will adapt to work together with other robots that currently are still able to operate, there is no single source of system failure, which is the case when there is a central computer controlling the system. Another advantage of not needing a central server is the ability to operate underground or in faraway places without a continuous radio link, which can be difficult under the previously mentioned circumstances.

A crucial part of such a redundancy-based system, where there are no single sources of failure, is to have backup sensor equipment. In the case of conflicts between separate sensor readings that should have matched, agents can negotiate among themselves to decide on the action to take to resolve the discrepancy. For example, if a teacup falls to the floor, and the audio sensor is broken, the fact that the video and image processing equipment registered the fall of the teacup will result in a negotiation session. The teacup fell according to the agent controlling video analysis, but the audio analyzer determined that the teacup did not fall—there was no sound of the shattering cup. In these cases, the two agents most likely will conclude the teacup did fall in the end, especially if the audio agent is capable of realizing something may be wrong with its sensors due to the video backup. Or the agents together can determine if further detail is required and ask an agent in control of a small reconnaissance robot to move to the projected site where the teacup fell and inspect the floor for cup fragments. The system will still be able to determine the need to order new teacups, even though the audio sensor that usually determines the need for new teacups currently is broken. This example displays one of the primary research motivations for multi-agent systems and agent frameworks—the ability to continue operation even if parts of the system are damaged or unavailable. This aspect is in sharp contrast to the usual state of affairs in the world of computer science; for example, even changing a single bit in a stream of code of a word processor program usually will break it to the point that it will not function at all.

Another generally less important but still significant motivation for MAS research is the potential benefit of using it as a basis for systems that exhibit emergent behavior. Emergent behavior refers to complex behavior of a system of many agents, even though none of the individual components (agents) has any kind of complex code. Emergent behavior is functionally equivalent to the relatively complex workings of a colony of ants capable of feeding the colony, relocating the hive when needed, and fending off predators, even though a single ant is not endowed at all with any kind of advanced brain function. More specifically, ants always will dispose of dead ants at the point that is farthest away from all colony entrances. A single ant clearly cannot solve this relatively complex geometrical problem; even a human being needs mathematical training before being able to solve such a geometric problem. The ability to find the answer to the problem of finding the farthest point from a set of points is an emergent ability displayed by ant colonies. The goal of emergent behavior research is to create systems that are robust in doing a very complex job, even with very simple equipment, contrasted to products that are clunky to use, hard to maintain, and require expensive equipment, as created by traditional programming styles. Areas where emergent behavior has proven to work can be found first and foremost in nature: Intelligence is evidently an emergent property; a single brain cell is government by extremely simple rules, whereas a brain is the most complex computer system known to humankind. This example also highlights the main problem with emergent behavior research; predicting what, if any, emergent behavior will occur is almost impossible.