

## **Introduction**

The term autonomous reflects some form of intelligence, whereas automatic generally signifies the lack of it. The term agent connotes mobility and the ability to leave one environment and enter another. Generally, an autonomous agent means a traveling intelligent entity that may interact with other entities in the environments that it visits.

The research on the theory of autonomous agents will expedite once there is a means to create actual solutions and perform measurable experiments. Indeed, the existence of an infrastructure for experimentation is a prerequisite for theoretical research in an area involving a high degree of complexity and novelty.

A researcher attempting an algorithmic solution for a particular class of problems is not concerned with how someone will implement his algorithm in some language. Nevertheless, it is the researcher's ultimate hope that someone will in fact demonstrate a practical verification of his theory.

This note discusses the significance of Z++ formalism in creating actual software based on theoretical ideas on autonomous agents.

### **The Z++ engine**

An agent enters an environment to gather data and possibly perform some tasks. This implies that the agent must be able to execute its code and also interact with other agents or applications in the environment. An agent may also have to signal its presence to other associated agents upon arrival.

The domain of operation of an agent stretches all the way down to middleware, and at times lower. Thus, the formalism for crafting agents requires an engine quite apart from a virtual machine augmented with proxies or XML.

An infrastructure for autonomous agents has been among the goals of Z++ research from the very beginning. An entirely new approach was necessary to build an engine to support an extensive formalism, on the wide range of domains and platforms. The approach became the genesis of Z++ Virtual Processor.

The Z++ Virtual Processor eliminates the need for proxies, data transformations, indeed everything beyond a direct algorithmic implementation of the research idea.

### **The Z++ formalism**

In terms of software, the notion of formalism is effectively a synonym for an abstract programming language. Of least interest to a researcher are awkward tricks specific to a particular language, or the target platform. On the other hand, useful general techniques within a formalism but not specific to any tool or platform are normal. The Z++ Virtual

Processor guarantees the uniqueness of the abstract semantics of statements on all concrete platforms.

Z++ formalism includes the linguistic facilities of C++, APL, ADA and Eiffel, coherently designed as a superset of C++. In Z++ formalism, agent communication, local or remote, is performed via the familiar syntax of invoking a method on an object. Agents can also communicate through user-defined signals and events.

### **Conclusion**

Almost any apparatus or remote mechanism becomes considerably more useful when equipped with the means that facilitate the deployment of intelligent mobile agents. Indeed, agents are the most fascinating means of extending our capabilities in intelligent ways. Furthermore, autonomous agents written in Z++ can penetrate the middleware layer, thereby extending the scope of research in this exciting area of limitless possibilities.

The distance between researchers and the implementers has impeded the progress of autonomous agents. Z++ formalism provides direct linguistic support for communicating mobile agents and component-oriented development. Researchers can design their prototypes in an expressive formalism without having to resort to peculiar programming tricks, or mixing multiple languages.

Z++ formalism is the medium for lasting abstract software.