SodaBot: A Software Agent Environment and Construction System

Michael H. Coen*
MIT Artificial Intelligence Laboratory
545 Technology Sq. NE43-823
Cambridge, MA 02139
mhcoen@ai.mit.edu

Much of the work done in the area of software agents can be placed into one of two categories: (1) highly theoretical treatment of agents' intentions and capabilities; and (2) applied construction of specific agents. However, determining for what (and if) software agents are actually useful requires building many of them, and the agent construction process poses difficult technical challenges.

Building agents generally involves a multi-layered approach ranging from low-level "system-hacking" (e.g. of mailers, networks, etc.) to high-level application development (e.g., a meeting scheduler) and everything in between. Each of these layers can require a substantial amount of independent implementation and debugging time. Additionally, it can be difficult to distribute new agents; they tend to be site-specific in intricate ways and disconnecting them from their local dependencies can be technically involved; for similar reasons, they can be difficult to install.

This abstract describes SodaBot, a general-purpose software agent user-environment and construction system. In SodaBot, each user is given a personal basic software agent (BSA) which typically runs in the background on her home workstation. The BSA is an agent operating system. By this, we mean that it is a generic (in the sense of universal) computational framework for implementing and running specific agent applications. The BSA is programmed in the SodaBot agent programming language (SodaBotL). As a quick sanity check, see if the following (rough) analogy makes sense:

SodaBotL is to SodaBot the way C Language is to Unix.

A BSA simultaneously (via a time-sharing algorithm) runs multiple SodaBotL programs provided both by its owner and by other people. The BSA architecture disconnects agent programs from the specific computational environment in which they run. They no longer need to be "hard-coded" with specific parameters for particular activities.

The SodaBot agent programming language (SodaBotL) offers high-level primitives and control-structures designed around human-level descriptions of agent activity. In SodaBotL, users can easily implement a wide-range of typical software agent applications, e.g. personal on-line assistants and meeting scheduling agents. SodaBotL abstracts out the low-level details of agent implementation. In a typical Unix environment, for example, agent creators are freed from the bother of dealing with system calls, mail servers, sockets, and X-windows. It is therefore much easier, for example, to have an agent:

- Interact with the user:
  
  Get Response {prompt "And what's your opinion?"; timeout in 10 minutes}
  $pollster_query;

- Handle time:
  
  Wait until Tuesday before $date:
  Display "Remind you have an appointment with $person on $date";

Additional features of SodaBot include automatic distribution of user-created agents and a graphical user interface. SodaBot has been built and tested, and it is in current development and use at the MIT AI Lab.

*The research described here was conducted at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124.

1Pronounced “Soda-Bottle.”

Figure 1: The SodaBot Basic Software Agent architecture