First Implementation of the World-Wide-Mind

Ray Walshe [1] and Mark Humphrys [2]

Dublin City University, School of Computer Applications,
Glasnevin, Dublin 9, Ireland.
[1] *ray@compapp.dcu.ie*, www.compapp.dcu.ie/~ray
[2] *humphrys@compapp.dcu.ie*, www.compapp.dcu.ie/~humphrys

Poster in *Proc. 6th European Conf. on Artificial Life (ECAL-01)*.

Abstract. The "World-Wide-Mind" (WWM) is introduced in [1]. For a short introduction see [2]. Under this scheme, it is proposed that autonomous agents researchers in AI and ALife construct their agent minds and agent worlds as *servers* on the Internet. Users will be able to run remote 3rd party Minds in other remote 3rd party Worlds. And users will be able to construct complex "Societies of Mind" out of many different remote Mind servers, and run this Society as a single Mind in some World. The motivation is: (a) to re-use other people's Worlds, (b) to re-use other people's Minds as components in larger, multiple-mind cognitive systems, and (c) to divide up the work in AI, so people can specialise on different parts.
This poster details the first working implementation of this idea. The key principle behind this implementation is to make it *as trivially easy as possible* for any author of a World or Mind to put it online without having to learn any network programming or any particular language. All technology involving a particular programming language (e.g. Java) or requiring the learning of particular programming skills (e.g. sockets) has therefore been rejected. A solution is proposed where the Mind or World author need only know how to repeatedly read plain text from and write plain text to a local file.

1 Introduction

Using the terminology in [1], a "World" server is a general environment that can be queried for the current "state", and that receives "actions" to be executed in that World. A "Mind" server receives state and returns an action. [1, §9] attempts to define a full set of queries that WWM servers should respond to. The lowest common denominator approach explained in [1, §11] argues that all queries to a server and all response data should be plain text in a standard, extensible XML format.

2 Putting a Mind or World on-line

Given that a researcher in ALife (or related fields) has constructed a Mind or World, how can this be put on-line? We propose that the Mind or World be a program running on a Web server, repeatedly reading and writing local files. A separate program will then provide on-line access to these files. For both Minds and Worlds, the Web server needs installed:

1. **Program 1 - The WWM server - A persistent program, representing the Mind or World.** - This will repeatedly read plaintext queries from and write plaintext responses to local files on the server. It will have its own persistent data structures in memory (global variables, gridworld, state-space, neural network, etc.). It doesn't have to be repeatedly launched. It launches once, and then repeatedly reads queries from the file until (if ever) the query "End run" is read.
2. **Program 2 - The CGI script - A non-persistent program through which the outside world talks to the persistent program.** - This accepts a WWM command across CGI, reads or writes a local file on the server, and exits. The "Start run" command should start a persistent instance of Program 1.

Now, to put his Mind or World online, the server author only has to write Program 1 (i.e. rewrite his program so it repeatedly reads/writes a local file). He may download a standard Program 2 from other WWM researchers. He then installs Program 2 in `cgi-bin` (or wherever) on his Web server, and then installs his Program 1 in some directory where the CGI script can run it from.

3 Program 1

Pseudo-Code examples are now provided. Here, the XML queries are not yet implemented. The World simply repeatedly reads actions and outputs state, and the Mind vice-versa. An initial system like this is running on our first node: `wwm.compapp.dcu.ie` with CGI scripts in Perl and WWM servers in Java. Obviously it must be shown that WWM servers can be written in any language.

3.1 The original AI program

The original AI program is probably an offline, combined World and Mind, program something like the following:

```
repeat
{
 State = World.GetState();
 Action = Mind.GetAction(State);
 World.ExecuteAction(Action);
 State = World.GetState();
}
```

3.2 The re-written World server

To put his system online, the AI author will break it into two programs, a World server and a Mind server. The World server will look something like this:

```
// Set up initial state, so as to provoke initial action:
State = GetState();
WriteStateFile(State);

repeat
{
 Action = ReadActionFile();

 if (Action != null)
 {
   ExecuteAction(Action);
   State = GetState();
   WriteStateFile(State);
 }
}
```

There is a StateFile and an ActionFile on the World server.

3.3 The re-written Mind server

```
repeat
{
 State = ReadStateFile();

 if (State != null)
 {
   Action = GetAction(State);
   WriteActionFile(Action);
 }
}
```

There is also a StateFile and an ActionFile on the Mind server. Remember this is a different machine, so these are different files to the ones the World server sees.

4 Program 2

Remember, Program 2 is simply downloaded from a standard source.

4.1 World CGI script

```
if (Query == StartRun)
        run_command(StartWorld);
if (Query == EndRun)
        run_command(EndWorld);
if (Query == GetState)
{
        State = ReadStateFile();
        echo State
}
if (Query == ExecuteAction)
{
        get Action from QUERY_STRING
        WriteActionFile(Action);
}
```

There are two basic commands - `GetState()` is how the states that the World is writing to the StateFile are read by the outside world - and `ExecuteAction(Action)` is how Actions from the outside world get into the ActionFile for the World to read.

4.2 Mind CGI script

```
if (Query == StartRun)
        run_command(StartMind);
if (Query == EndRun)
        run_command(EndMind);
if (Query == HereIsState)
{
        get State from QUERY_STRING
        WriteStateFile(State);
}
if (Query == GetAction)
{
        Action = ReadActionFile();
        echo Action
}
```

The single basic command `GetAction(State)` could be broken into *two commands* - one that tells the Mind the state, and another that collects the Action from the ActionFile later.

5 Conclusion

We have demonstrated a lowest-common-denominator approach to implementing the WWM concept. We believe considerable work needs to be done on making installation as simple as possible in any language on any OS for this scheme to become widely used. We are working on defining rudimentary "template" Minds and Worlds in all major programming languages to show mind and world authors how to get them online. Funding has also been secured to work on client user software.

References

1. Humphrys, M. (2001), *The World-Wide-Mind: Draft Proposal*, Dublin City University, School of Computer Applications, Technical Report CA-0301, www.compapp.dcu.ie/~humphrys/WWM
2. Humphrys, M. (2001a), Distributing a Mind on the Internet: The World-Wide-Mind, *ECAL-01*.