Emergent Behaviours Considered Harmful

Dana Moore and William Wright
BBN
1300 N 17th Street
Arlington, Virginia 22209
703-284-4600
[damoore ,wwright]@bbn.com

ABSTRACT

Autonomous Agent Systems have proven difficult to instrument, monitor, control and test. Unit testing strategies are of little help, as they tend to stress functional aspects of a system rather than modal aspects and test at too low a level (the unit rather than the system). Further, emergent behaviours, indeterminacy, and the loose coupling of system elements often obscure the system's attributes. Absent specific inclusion in the system design of features that provide such specific capabilities, developers have little recourse to understand the systems they build. In this paper we describe ACME, a parallel agent system for testing used in UltraLog, a DARPA effort aimed at creating an ultra-survivable multi-agent system. We describe the motivation and architecture for a comprehensive testing system, and suggest a path for other MAS projects to improve testing and validation of their systems. Finally, we discuss the need for ad hoc logic changes in a running MAS, an instant messaging strategy for conversing with the system (including facilities for AliceBot and Jabber). We describe the use of the Ruby scripting language in the system and summarize the benefits and lessons learned.

Categories and Subject Descriptors

D.2 [**Software Engineering**]:
D.2.3 Coding Tools and Techniques – *Standards*
D.2.4 Software/Program Verification -- *Reliability*
D.2.5 Testing and Debugging – *Debugging aids, Distributed debugging, Testing tools*

General Terms

Measurement, Experimentation, Verification.

Keywords

Instrumentation, Verification, Validation, Testing, Controls, Agent-based Software Engineering; Agent-based Workflow Management. Testing Methodologies, Validation Methodologies.

1. INTRODUCTION

Emergent behaviours, defined as expected or unpredicted events produced by the complex asynchronous interaction between agents in a MAS, are likely to occur. Such behaviours will be at times beneficial and hoped and at time unpredicted and unwanted.

AAMAS'03, July 14–18, 2003, Melbourne, Australia.
ACM 1-58113-683-8/03/0007.

In DARPA's UltraLog program [5], where research is ongoing into creating and measuring the survivability of very large MAS, emergent behaviours, perturbation and events often evince themselves and do not necessarily represent degenerate cases. UltraLog in fact relies upon coordinative and potentially emergent properties to assure continued operations in the face of cyber or kinetic attacks.

Our ongoing mission is (under the control of a testing framework) to:

- Inject cyber and kinetic stresses into the running system,
- Stimulate defensive behaviours,
- Recognize responses as they occur, characterize them,
- Measure their effect on the system of agents and their progress toward solving a problem given to the system, and finally
- Associate those effects with the components of the system and use those measurements to improve the appropriate components.

DARPA's UltraLog program relies upon components independently designed and built by scores of software developers. More often than not, developers create agents which are narrowly focused and self-interested. Even though multiple redundant mechanisms -- playbooks for case based responses, technical specifications to describe agents, policy binders, community-of-interest managers -- exist to govern interactions, there is *still* no question of whether the unexpected will happen. The interesting questions are really more whether observers of the system (including both human and automated testing systems) can:

- Perceive events of interest,
- Understand and react to such events to coax meaningful results from them,
- Stimulate the emergence of certain arbitrary behaviours and possibly control or orchestrate them, and finally
- Produce meaningful and human understandable conclusions about a string of experiments.

2. MOTIVATIONS

The core problem in realizing the vision of any non-trivial MAS is to create well-behaved systems that tolerate fault and error, provide desired functional and modal characteristics. Emergent behaviours and reaction to perturbation and stress, although

unpredictable, should not be functionally devastating. The extent to which a design and implementation meets these (and related) challenges defines it as a well-behaved system.

3. THE COUGAAR AGENT SYSTEM

COUGAAR is an agent architecture well suited to modeling workflows, superior-subordinate and supplier-consumer relationships. Table 1 depicts a hierarchical model of COUGAAR as it implements UltraLog.

Table 1 - COUGAAR System Hierarchy

Level Of Hierarchy…	Functionality Encapsulated at this Level…
Plugin	Business logic, agent "personality".
Agent	Container supplying life-cycles services.
Community	A neighbourhood of agents providing community of interest and facilities for multi-agent adaptation.
Society	A set of agents operating together as a (potentially very large) distributed "black box".

4. TECHNICAL APPROACH TO MAS ASSURANCE

While the practice of code design, development and inspection practice is well understood, the grounds for testing and evaluating loosely coupled systems in general (and MAS in specific) is lightly trodden territory. Here, we enumerate basic elements of our approach.

- Dashboarding: Using the build and evaluation tools on an hourly basis, we connected their output to a graphical project 'dashboard' or Executive Information System fed by the check-in and build process [6].
- Code Instrumentation: We use a simple approach that we believe is generally applicable: mandating the use of structured Log4J [2] messages to provide first order code instrumentation and simple mediators (called 'appenders' in Log4J parlance) that would enable event sensing and response without demanding additional intense developer focus. Event sensing and response are linchpins in creating an experiment automation framework. Accordingly, we were able provide instrumentation (dip clips) onto the runnable code, with significant capabilities.
- Test Automation: Our basic premise is that, *an agent-based system might be the most appropriate way to test an agent-based system*. For this purpose, we developed a separate agent system, called ACME.

4.1 ACME: Toward a 'Dip-Clip' Approach

Much in the same way that COUGAAR and other agent frameworks support 'plugin' of new capabilities, we felt that a general ability to do the same, to in effect create an agent-based testing framework would be required to support the rapid development of test suites that could be applied to MAS testing. Therefore we designed and implemented ACME to support several features that our experience suggested were essential. We knew that the ability to deliver a new service that would exist either as a part of the COUGAAR agent society or on in-society hosts would be needed.

4.1.1 ACME architecture

ACME was built using Ruby [4], an interpreted object-oriented agile language with excellent support for system management and native platform integration and is platform agnostic. ACME's service architecture is built on an open source component architecture [3], and uses plugin agents (called botlets) to provide specific services for System-to-system interactions (handled through instant messaging using Jabber [1], an extended XML messaging protocol. To provide a human-to-system interface without special programming, we commonly use one of several freely available Jabber IM clients.

5. CONCLUSIONS

Best current practice in component testing will not reveal the dynamical behaviours in a system such as UltraLog. Our experience creating this testing system suggests that frameworks for challenging and evaluating societies are an important element of the MAS rather than an afterthought. Our framework has allowed us to build, exercise, and evaluate an extremely complex MAS using a suite of interactive tools as well as another MAS for automated testing and evaluation. The tools that we have developed in pursuit of survivability goals point toward a general method for controlling and interacting with agent societies.

6. ACKNOWLEDGEMENTS

This work is sponsored by the US Defense Advanced Research Projects Agency (DARPA) and is managed under DARPA's Joint Logistics Technology Office (JLTO).

7. REFERENCES

[1] Jabber Project Homepage: www.jabber.org

[2] Jakarta Log4J Project Homepage: http://jakarta.apache.org/log4j/docs/

[3] FreeBase Homepage http://www.rubyide.org/cgi-bin/wiki.pl?FreeBASE

[4] Ruby Language Homepage: http://ruby-lang.org/en/index.html

[5] UltraLog Project Homepage: http://www.ultralog.net

[6] Ultralog Project Management Dashboard: http://cvs.ultralog.net