Abstract. Advances in Agent Oriented Software Engineering have focused on the provision of frameworks and toolkits to aid in the creation of Multi Agent Systems (MASs). However, despite the need to address the inherent complexity of such systems, little progress has been made in the development of tools to allow for the debugging and understanding of their inner workings.

This paper introduces a novel performance analysis system, named AgentSpotter, which facilitates such analysis. AgentSpotter was developed by mapping conventional profiling concepts to the domain of MASs. We outline its integration into the Agent Factory multi agent framework.

1 Introduction

Recent developments in the area of Multi Agent Systems (MASs) have been concerned with bridging the gap between theory and practice, by allowing concrete implementations of theoretical foundations to be built and deployed. However, the dearth of agent-specific development and debugging tools remains a significant obstacle to MASs being adopted in industry on a large scale.

While some simple debugging and logging tools exist for MAS analysis, these tend not to aid in reasoning about large-scale system when viewed at the high agent-oriented abstraction level. Such tools typically allow for traditional debugging actions such as state stepping and breakpoint insertion.

One popular performance analysis technique is known as profiling. Profiling is based on the observation that the majority of the execution time of a program can be attributed to a small number of bottlenecks (or hot spots). By improving the efficiency of these portions of a program, overall performance can be dramatically improved. Profiling was initially introduced by Donald E. Knuth in an empirical study conducted on FORTRAN programs \[\text{1}\]. Since then, the technique has been successfully applied to a variety of languages, platforms and architectures.

The aim of this paper is to apply the principles of traditional profiling systems in a multi agent environment, so as to facilitate the developers of MASs in debugging their applications by gaining a better understanding of where the bottlenecks exist and performance penalties are incurred.
This paper is organised as follows: Section 2 provides a brief overview of existing tools aimed at aiding in the analysis of MASs. In Section 3 we introduce the AgentSpotter profiling system, with particular focus on outlining a conceptual model for generic MAS profiling. A concrete implementation of this work, aimed at the Agent Factory MAS framework, is outlined in Section 4. Section 5 presents the space-time diagram produced by AgentSpotter in more detail, with an evaluation of its usefulness given in Section 6. Finally, Section 7 presents our conclusions and ideas for future work.

2 Related Work

In designing a profiler for MASs, the features that tend to be present in traditional profilers for non-MAS applications must be identified. It is also necessary to examine those debugging and analysis tools that already exist for MASs.

The motivation behind the use of profiling on computer applications is clearly outlined in Knuth’s observation that “less than 4% of a program accounts for more than half of its running time” [1]. This statement implies that a developer can achieve substantial increases in performance by identifying and improving those parts of the program that account for the majority of the execution time. The key aim of profilers is to identify these bottlenecks.

Another observation leading to the widespread adoption of profilers as debugging tools is that there frequently exists a mismatch between the actual run-time behaviour of a system and the programmers’ mental map of what they expect this behaviour to be. Profilers are useful in enlightening developers to particular aspects of their programs that they may not otherwise have considered.

A traditional profiler typically consists of two logical parts. Firstly, an instrumentation apparatus is directly weaved into the program under study or run side-by-side to gather and record execution data. Secondly, a post-processing system uses this data to generate meaningful performance analysis listings or visualisations.

In the traditional software engineering community, historical profilers such as gprof [2] or performance analysis APIs like ATOM [3] and the Java Virtual Machine Tool Interface (JVMTI) [4] have made performance analysis more accessible for researchers and software engineers. However, the MAS community does not yet have general access to these types of tools.

Unique amongst all of the mainstream MAS development platforms, Cougaar is alone in integrating a performance measurement infrastructure directly into the system architecture [5]. Although this is not applicable to other platforms, it does provide a good insight into the features that MAS developers could reasonably expect from any performance measurement application. The principal characteristics of this structure are as follows:

– Primary data channels consist of raw polling sensors at the heart of the system execution engine that gather simple low-impact data elements such as counters and event sensors.
– Secondary channels provide more elaborate information, such as summaries of the state of individual components and history analysis that stores performance data over lengthy running times.