Demo paper: AGADE
Scalability of ontology based agent simulations

Thomas Farrenkopf[1], Michael Guckert[1], Neil Urquhart[2] and Simon Wells[2]

[1] KITE - Kompetenzzentrum für Informationstechnologie, Technische Hochschule Mittelhessen, Germany
`{thomas.farrenkopf, michael.guckert}@mnd.thm.de`
[2] School of Computing, Edinburgh Napier University, Scotland
`{n.urquhart, s.wells}@napier.ac.uk`

Abstract. Simulations of real world scenarios often require considerably large numbers of agents. With increasing level of detail and resolution in the underlying models machine limitations both in the aspect of memory and computing power are reached. Even more when additional features like reasoning mechanisms of semantic technologies are used as in the AGADE framework where we have extended the principal BDI paradigm with an interface to OWL ontologies. We have observed that the extensive use of ontologies results in high memory consumption due to the large number of `String` objects used in the reasoning process and caching mechanisms of the OWL API. We address this issue by running simulations in a highly distributed environment. In this paper we demonstrate how we enabled AGADE to be run in such an environment and the necessary architectural modifications. Furthermore, we discuss the potential size of simulations that can be run in such a setting.

Keywords: Multi-Agent System, BDI, OWL Ontology, Market Simulation, Distributed Simulation

1 Introduction

Using OWL ontologies [7] in a usual single instance setting memory consumption is not a serious issue. But using ontologies in multi-agent simulations where each agent is equipped with its own ontology memory usage increases linearly due to the extensive use of `String` objects used in the reasoning process and caching mechanisms of the OWL API [4]. As AGADE keeps representations of social relations in the ontology the effect is enforced with increasing number of agents and consequently rising numbers of mutual connections. AGADE uses ontologies to flexibly model world knowledge in a BDI architecture. See [1] for detailed examples of how ontologies can be integrated into the BDI concept in general and the Jadex [6] framework in particular. Although, distributed agent platforms have long been available, but the support of crucial elements of multi-agent simulations e.g. schedulers for managing time, synchronising agents and data collection is often not available [5]. Therefore AGADE been implemented in a non distributed environment at first and has then been extended. Multi-agent simulations can now be run in a network and offer full support of a model of time and elaborate communication mechanisms so that scenarios can be scaled up and run in the network or not.

2 Main purpose

AGADE is a BDI based framework that supports the development of dynamic business scenarios with individual access to semantic reasoning for each agent [2]. AGADE agents are equipped with their own inference engine (reasoner) and private ontology using the OWL API [4]. The ontology allows a flexible architecture as aspects of the agent may be expressed in the rules rather than static code [1]. We distinguish between the abstract domain layer (ADL) containing general knowledge and the specific domain layer (SDL) shared by all agents and the individual domain layer (IDL) specific to each agent. AGADE allows scaling up the amount of agents participating in a simulation scenario using a directing agent that orchestrates the set of agents and controls the flow of time and communication. Each round (time step) of the simulation is structured into four phases (calculation, socialisation, action, control) with defined functionality and integration into the BDI paradigm.

3 Demonstration

AGADE implements a Java RMI (Remote Method Invocation) based communication mechanism with which the agents can send and receive messages. RMI is directly based on socket communication and is therefore more efficient than alternative technologies like Web Services or CORBA. We ran various benchmarks comparing the alternatives with results similar to what has been published before (e.g. [3]). Consequently, we decided to use RMI which is currently still the best technology for running distributed tasks in the context of multi-threaded applications in Java.

On an abstract level AGADE knows two different kinds of BDI agents: a single director type agent that acts as controlling instance of the simulation and performs central administrative tasks (e.g. controlling simulation rounds by triggering each agent), and a participant type agent that acts in the simulation (i.e. consumer or seller). The communication between participating agents and between participating agent and director agent uses RMI based services both in distributed and in non-distributed mode ensuring a unified architecture for the framework.

The typical RMI flow of execution starts with an initial registration of objects in the RMI registry of a server using a unique name which makes the objects available for client access. Clients can now query the lookup service of the *RMI registry* to get a reference to the objects. The client can then invoke appropriately published methods. In our AGADE implementation the role of server and client are interchangeable i.e. each node in the network can act as client and as server thus allowing two way communication with asynchronous method invocation (see Fig. 1). AGADE implements a `YellowPages` service in each node where agents and their node are published to be used in inter-agent communication. Note that two nodes in the network do not necessarily have to be connected through yellow page entries as they may not have to communicate at all during the simulation. But at least one distinguished central node must be aware of all other nodes and can then act as a broker and can enable communication if requested. Once communication between two nodes has been established mutual entries are made in the yellow pages. This lazy set up of communication information reduces initial messaging efforts and provides direct links only if requested. The director agent and the GUI communication with the user obviously have to be placed on the central node as well. The delivery process of inter-agent communication is therefore implemented as follows:

– query local yellow pages on client-side and deliver message directly if both address information is available
– request broker on central node to provide necessary information and initiate communication

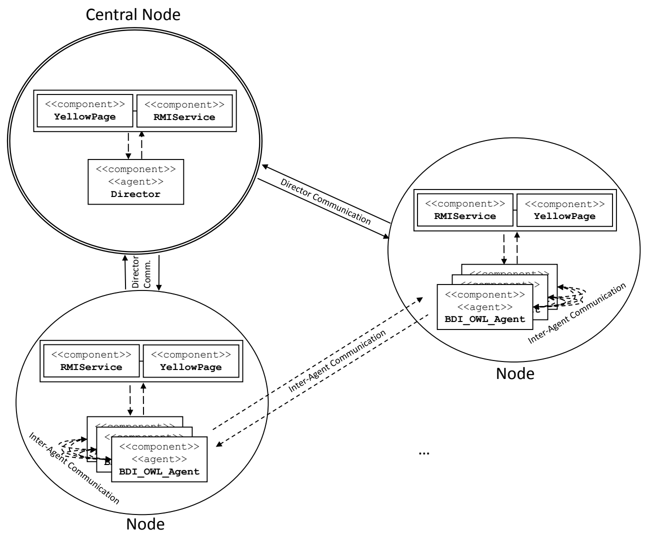

Fig. 1. AGADE nodes model.

In the inter-client communication scenario, the client is querying the server for the client-address and will then deliver the message to the specific client directly. RMI message mechanisms use Java serialisation to deliver objects. OWL components are represented as Strings and can therefore undergo the standard RMI serialisation process. Thus the content of each agent message is eventually a serialised instance of `String`. The typical message content size of an OWL concept is about 850 Byte = 0.0068 Mbit. Theoretically, a typical network bandwidth of 1000MBit/s allows sending $\sim 147,058$ messages simultaneously with respect to the content size without limitations neglecting connection overhead usage. To reduce the amount of network traffic between clients, we further compress the message content by using *gzip* which reduces the size by roughly 50 percent.

This architecture allows to scale up simulations considerably. We successfully tested the robustness of this architecture by simulating a homogeneous crowd of buyers acting in a mobile phone market where we have run 100,000 agents over 100 rounds on 6 clients each connected with 100 MBits network speed and equipped with various hardware settings (RAM/CPU).

4 Conclusion

In this demonstration paper, we have presented an improved version of AGADE, where round-based multi-agent simulations with elaborate agent behaviour modelled in OWL can be scaled up considerably. We addressed the high memory consumption of the OWL reasoning process by running simulations in a distributed environment where full support of crucial characteristics of MAS (e.g. well defined phases, common model of time,...) are guaranteed. As a result, AGADE can now handle simulation scenarios with a much higher number of agents in a single architecture that can easily be configured to be run on a single machine or in a network with distributed nodes.

References

1. Farrenkopf, T., Guckert, M., Hoffmann, B., Urquhart, N.: Multiagent System Technologies: 12th German Conference, MATES 2014, Stuttgart, Germany, September 23-25, 2014. Proceedings, chap. AGADE How Individual Guidance Leads to Group Behaviour and How This Can Be Simulated, pp. 234–250. Springer International Publishing, Cham (2014), `http://dx.doi.org/10.1007/978-3-319-11584-9_16`
2. Farrenkopf, T., Guckert, M., Urquhart, N.: Advances in Practical Applications of Agents, Multi-Agent Systems, and Sustainability: The PAAMS Collection: 13th International Conference, PAAMS 2015, Salamanca, Spain, June 3-4, 2015, Proceedings, chap. AGADE Using Personal Preferences and World Knowledge to Model Agent Behaviour, pp. 93–106. Springer International Publishing, Cham (2015), `http://dx.doi.org/10.1007/978-3-319-18944-4_8`
3. Gray, N.A.B.: Comparison of web services, java-rmi, and corba service implementation. In: Fifth Australasian Workshop on Software and System Architectures (2004)
4. Horridge, M., Bechhofer, S.: The owl api: A java api for owl ontologies. Semantic Web 2(1), 11–21 (2011)
5. Mengistu, D., Troger, P., Lundberg, L., Davidsson, P.: Scalability in distributed multi-agent based simulations: The jade case. Future Generation Communication and Networking Symposia, International Conference on 5, 93–99 (2008)
6. Pokahr, A., Braubach, L., Jander, K.: The jadex project: Programming model. In: Ganzha, M., Jain, L.C. (eds.) Multiagent Systems and Applications, Intelligent Systems Reference Library, vol. 45, pp. 21–53. Springer Berlin Heidelberg (2013)
7. W3C OWL Working Group: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (11 December 2012), available at `http://www.w3.org/TR/2012/REC-owl2-overview-20121211/`